(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－265283

(43)公開日　平成11年(1999)９月28日

| (51)Int.Cl.[6] | | 識別記号 | FI | | |
|---|---|---|---|---|---|
| Ｇ０６Ｆ | 9/06 | ５４０ | Ｇ０６Ｆ | 9/06 | ５４０Ｅ |
| | | | | | ５４０Ｍ |
| | 9/22 | ３７０ | | 9/22 | ３７０ |
| | 12/06 | ５２０ | | 12/06 | ５２０Ｅ |

審査請求　未請求　請求項の数５　ＯＬ　（全 12 頁）

(21)出願番号　特願平10－68102

(22)出願日　平成10年(1998)３月18日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　田村　隆之
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　片山　国弘
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

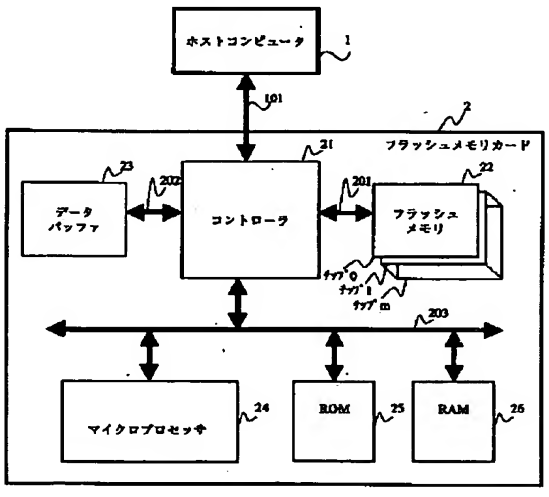
(72)発明者　中村　一男
東京都小平市上水本町五丁目20番地１号
株式会社日立製作所半導体事業部内

(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　記憶装置におけるファームウェアの修正方法及び記憶装置

(57)【要約】
【課題】フラッシュメモリなどの不揮発性半導体メモリを用いた記憶装置において、内蔵されるファームウェアの容易な修正と低価格な記憶装置を提供する。
【解決手段】フラッシュメモリカード２において、電源オン時に、フラッシュメモリ２２からＲＡＭ２６にロードすることで、ＲＯＭ２５内の不具合が発生したモジュールの変わりにＲＡＭ２６の不具合を修正されたモジュールを使用すること、ＲＯＭ２５のファームウェアの不具合対策を行う。

図1



【特許請求の範囲】
【請求項１】ホストコンピュータが書き込むデータを格納するための不揮発性半導体メモリと、ファームウェアが格納されている第１のメモリと、ファームウェアを処理するためのマイクロプロセッサを有する記憶装置において、該第１のメモリに格納されているファームウェアに不具合が発生したときのファームウェアの修正方法であって、該第１のメモリに格納されているファームウェアに不具合が発生したときには、該不揮発性半導体メモリに不具合が発生したファームウェアのモジュール、および不具合が発生したモジュールとの関係を示すジャンプテーブルが格納され、電源オン処理において、該不揮発性半導体メモリに該修正されたモジュールとジャンプテーブルが格納されている場合には、該不揮発性半導体メモリから第２のメモリに修正されたファームウェアのモジュールとジャンプテーブルをロードし、該マイクロプロセッサは、該第２のメモリにロードされたジャンプテーブルを参照することで、該第１のメモリに格納されている不具合が発生したモジュールの変わりに該第２のメモリにロードされている修正されたモジュールを実行することを特徴とする記憶装置におけるファームウェアの修正方法。
【請求項２】請求項１に記載の記憶装置におけるファームウェアの修正方法であって、電源オン処理において、該第１のメモリに格納されているジャンプテーブルを第２のメモリにコピーし、該不揮発性半導体メモリに修正されたファームウェアのモジュールが格納されていない場合には、該マイクロプロセッサは、該第２のメモリに格納されているジャンプテーブルを参照することで、該第１のメモリに格納されているファームウェアを実行し、該不揮発性半導体メモリに修正されたファームウェアのモジュールが格納されている場合には、該不揮発性半導体メモリから第２のメモリに修正されたファームウェアのモジュールとジャンプテーブルをロードし、このとき該第１のメモリからコピーされたジャンプテーブルを上書きし、該マイクロプロセッサは、該第２のメモリにロードされたジャンプテーブルを参照することで、該第１のメモリに格納されている不具合が発生したモジュールの変わりに該第２のメモリにロードされている修正されたモジュールを実行することを特徴とする記憶装置におけるファームウェアの修正方法。
【請求項３】ホストコンピュータが書き込むデータを格納するための不揮発性半導体メモリと、ファームウェアが格納されている第１のメモリと、ファームウェアを処理するためのマイクロプロセッサを有する記憶装置において、該第１のメモリに格納されているファームウェアに不具合が発生したときには、該不揮発性半導体メモリに不具合が発生したファームウェアのモジュール、および不具合が発生したモジュールとの関係を示すジャンプテーブルが格納され、該不揮発性半導体メモリに修正されたファームウェアのモジュールが格納されている場合には、該不揮発性半導体メモリから修正されたファームウェアのモジュールとジャンプテーブルをロードするための第２のメモリを有し、該第１のメモリに格納されている不具合が発生したモジュールの変わりに該第２のメモリに格納されている修正されたモジュールを実行することを特徴とする記憶装置。
【請求項４】ホストコンピュータが書き込むデータを格納するための不揮発性半導体メモリと、ファームウェアが格納されている第１のメモリと、ファームウェアを処理するためのマイクロプロセッサを有する記憶装置において、該第１のメモリに格納されているファームウェアに不具合が発生したときのファームウェアの修正方法であって、該第１のメモリに格納されているファームウェアに不具合が発生したときには、該不揮発性半導体メモリに不具合が発生したファームウェアのモジュール、および不具合が発生したモジュールとの関係を示すジャンプテーブルが格納され、電源オン処理において、該不揮発性半導体メモリに修正されたファームウェアのモジュールが格納されている場合には、該マイクロプロセッサは、該不揮発性半導体メモリに格納されているジャンプテーブルを参照するように切り替え、該マイクロプロセッサは、該不揮発性半導体メモリに格納されているジャンプテーブルを参照することで、該第１のメモリに格納されている不具合が発生したモジュールの変わりに該不揮発性半導体メモリに格納されている修正されたモジュールを実行することを特徴とする記憶装置におけるファームウェアの修正方法。
【請求項５】ホストコンピュータが書き込むデータを格納するための不揮発性半導体メモリと、ファームウェアが格納されているメモリと、ファームウェアを処理するためのマイクロプロセッサを有する記憶装置において、該ファームウェアに不具合が発生したときには、該不揮発性半導体メモリに不具合が発生したファームウェアのモジュール、および不具合が発生したモジュールとの関係を示すジャンプテーブルが格納され、該不揮発性半導体メモリに修正されたファームウェアのモジュールが格納されている場合には、該メモリに格納されている不具合が発生したモジュールの変わりに該不揮発性半導体メモリに格納されている修正されたモジュールを実行することを特徴とする記憶装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、電気的に書き換え可能な不揮発性半導体メモリを記憶媒体として用いた記憶装置に係り、特に、記憶装置に格納されたファームウェアの修正を容易に行うことを目的とした不揮発性半導体メモリを用いた記憶装置に関する。
【０００２】
【従来の技術】不揮発性半導体メモリを用いた記憶装置

として、フラッシュメモリを用いた記憶装置がある。フラッシュメモリは、データの書き込みを行う前に、データ書き込みを行う一つ以上のセクタから構成されるブロックを消去しなければならない。また、パーソナルコンピュータ（ＰＣ）などのホストコンピュータに接続された場合、ＰＣが発行するコマンドなどの解析を行い、ＰＣが要求するデータの読み出しや書き込みを制御する必要がある。

【０００３】このフラッシュメモリの制御やホストコンピュータ間の制御を行うために、記憶装置にはファームウェアが格納されている。このような記憶装置として、米国特許第５６０６６６０号公報がある。本従来例において、ファームウェアは不揮発性半導体メモリに格納され、ＲＡＭなどのメモリにロードされた後に、記憶装置に内蔵のマイクロプロセッサによって実行される。不揮発性半導体メモリにファームウェアが格納されていない場合には、ホストコンピュータからの特殊コマンドによって、ホストコンピュータが書き込むファームウェアを不揮発性半導体メモリに書き込む。

【０００４】

【発明が解決しようとする課題】米国特許第５６０６６６０号公報に示される従来の記憶装置は、ファームウェアを不揮発性半導体メモリに格納するための手段についてのものであり、ファームウェアに不具合が発生した場合におけるファームウェアの修正に関して配慮がされておらず、容易にファームウェアを修正できないという問題がある。さらに、従来の記憶装置では、不揮発性半導体メモリに格納されたファームウェアをＲＡＭに格納するために、ＲＡＭの容量が増大する問題がある。

【０００５】本発明の目的は、不揮発性半導体メモリにファームウェアを格納した後に、ファームウェアの不具合が発生した場合でも、ファームウェアの修正を容易に行うことにある。

【０００６】本発明の他の目的は、ファームウェアを修正するために使用されるメモリの容量を低減し、低価格な記憶装置を提供することにある。

【０００７】

【課題を解決するための手段】上記目的を達成するために、本発明の記憶装置に格納されるファームウェアは、各機能ごとに分割された複数のモジュールと、各モジュールの呼び出し毎に参照されるジャンプテーブルで構成される。ジャンプテーブルは、各モジュールのアドレスを示している。また、本発明の記憶装置は、不揮発性半導体メモリであるフラッシュメモリ、マイクロプロセッサ、ファームウェアが格納されるＲＯＭ、そしてＲＯＭに格納されているファームウェア内のジャンプテーブルまたは修正されたジャンプテーブルと修正されたモジュールをフラッシュメモリからロードするＲＡＭから構成される。フラッシュメモリには、ホストコンピュータが書き込むデータと、修正されたジャンプテーブルや複数のモジュールが格納される。ＲＯＭには、予めファームウェアが格納されている。

【０００８】まず、電源オン後、第１のメモリに格納されているジャンプテーブルが、マイクロプロセッサによって、ＲＯＭからＲＡＭにコピーされる。ファームウェアに不具合が発生していないときには、ＲＡＭにコピーされたジャンプテーブルは、すべてＲＯＭに格納されているファームウェアの各モジュールのアドレスを示している。

【０００９】ＲＯＭに格納されているファームウェアに不具合が発生した場合、フラッシュメモリには、不具合が発生したモジュールに対する修正したモジュールを格納する。同時に、ジャンプテーブルも、修正されたモジュールのアドレスを示すように修正され、フラッシュメモリに格納される。

【００１０】ＲＯＭに格納されているジャンプテーブルをＲＡＭにコピーした後、修正されたジャンプテーブルおよび複数のモジュールがフラッシュメモリに格納されている場合には、フラッシュメモリに格納されている修正されたジャンプテーブルおよび複数のモジュールはＲＡＭにロードされる。このとき、ＲＯＭからコピーされたジャンプテーブルは、フラッシュメモリに格納されている修正されたジャンプテーブルに書き替えられる。これにより、ＲＯＭに格納されている不具合が発生したモジュールは参照されず、ＲＡＭにロードされている修正されたモジュールが参照される。

【００１１】以上のように、不具合が発生したモジュールおよびジャンプテーブルを修正し、フラッシュメモリに格納することにより、ＲＯＭに格納されているファームウェアに不具合が発生した場合にも、容易にファームウェアの修正が可能になる。

【００１２】また、ファームウェアの修正のために必要になるＲＡＭの容量は、不具合が発生したモジュールとジャンプテーブルの容量で十分であるので、ＲＡＭ容量を低減でき、低価格な記憶装置を提供できる。

【００１３】

【発明の実施の形態】以下、本発明に係る不揮発性半導体メモリを用いた記憶装置の実施例について説明する。

【００１４】図１に、本発明の一実施例である記憶装置の構成をコンピュータの外部記憶装置への適用を例にとり示す。

【００１５】図１において、２が外部記憶装置であり、ホストコンピュータ１の外部記憶装置である。ホストコンピュータ１は、記憶装置に対してデータの格納や読み出しを行う装置であり、例えばパーソナルコンピュータ（ＰＣ）である。外部記憶装置２は、不揮発性半導体メモリとしてフラッシュメモリを用いていることからフラッシュメモリカードと呼ばれる。

【００１６】このフラッシュメモリカード２は、標準バス１０１を介してホストコンピュータ１からのコマンド

やデータの授受を行う。標準バス１０１には、ＰＣＭＣＩＡインタフェースやＳＣＳＩインタフェースなど様々なインタフェースがあり、外部記憶装置を必要とするホストコンピュータ１と間のプロトコルの取り決めを有するものであれば特に限定はない。

【００１７】フラッシュメモリカード２において、２１は、ホストコンピュータ１やフラッシュメモリ２２とのインタフェースを司り、マイクロプロセッサ２４の指示に従って、ホストコンピュータ１やフラッシュメモリ２２間のデータ転送を行うコントローラである。

【００１８】２２は、ホストコンピュータが書き込むデータを格納するためのフラッシュメモリであり、複数個のフラッシュメモリチップから構成される。フラッシュメモリ２２には、ホストコンピュータが取り扱うユーザデータや修正されたモジュールやジャンプテーブル、そしてフラッシュメモリに格納されているユーザデータを管理するための管理情報が格納されている。

【００１９】２３は、ホストコンピュータ１との間でデータの授受を行うとき、データを一時格納するためのデータバッファである。データバッファ２３は、マイクロプロセッサ２４がフラッシュメモリ２２に格納されているデータを読み出したり、マイクロプロセッサ２４がフラッシュメモリ２２にデータを書き込むときにも使用される。データバッファ２６は、スタティックＲＡＭやダイナミックＲＡＭそしてランダムアクセスが可能なフラッシュメモリなどのランダムアクセスが可能なメモリであればよい。

【００２０】２４は、マイクロプロセッサであり、ホストコンピュータ１が書き込むコマンドの制御やフラッシュメモリ２２の管理、そしてフラッシュメモリ２２とデータバッファ２３間やホストコンピュータ１とデータバッファ２３間のデータ転送を制御する。

【００２１】２５は、ファームウェアが格納されている不揮発性のメモリ（ＲＯＭ）である。ＲＯＭ２５は、マスクＲＯＭやＥＥＰＲＯＭそしてフラッシュメモリなどの不揮発性のメモリであればよい。

【００２２】２６は、フラッシュメモリ２２に格納されている修正されたモジュールやジャンプテーブルがロードされるランダムアクセスが可能なメモリ（ＲＡＭ）である。また、ＲＡＭ２６は、マイクロプロセッサ２４が使用するワークメモリとしても使用可能である。ＲＡＭ２６は、スタティックＲＡＭやダイナミックＲＡＭそしてランダムアクセスが可能なフラッシュメモリなどのメモリであればよい。

【００２３】ここで、フラッシュメモリ２２は、フラッシュバス２０１を介してコントローラ２１に接続される。フラッシュバス２０１は、フラッシュメモリ２２に依存するインタフェースを提供する。データバッファ２３は、データバッファバス２０２を介してコントローラ２１に接続される。データバッファバス２０２は、データバッファ２３に依存するインタフェースを提供する。コントローラ２１、マイクロプロセッサ２４、ＲＯＭ２５そしてＲＡＭ２６は、ローカルバス２０３を介して、それぞれが接続される。ローカルバス２０３は、マイクロプロセッサ２４に依存するインタフェースを提供する。

【００２４】図２に、フラッシュメモリ２２のチップ０の内部構成を示す。

【００２５】ブロックアドレス０～ｎ－１には、ユーザデータが格納される。ブロックアドレスｎには、修正されたファームウェアが格納される。ブロックアドレスｎ以降のブロックは、フラッシュメモリ２２を管理するための情報が格納される。また、それぞれのユーザデータ部は、ユーザデータ管理部を持ち、ユーザデータ部に対する状態などの管理情報を保持している。ユーザデータ管理部の使用方法に関しては省略する。

【００２６】修正されたファームウェアが格納されている修正ファームウェア格納部２２１は、修正されたジャンプテーブルや複数の修正されたモジュールが格納されている。ユーザデータ部と同様に、修正ファームウェア格納部２２１は修正ファーム管理部２２２を持つ。修正ファーム管理部２２２は、修正ファームウェア格納部２２１に修正されたファームウェアが格納されているかどうかを示す修正ファーム有無フラグなどの情報を保持している。ＲＯＭ２５に格納されているファームウェアに不具合が発生していない場合には、修正ファーム有無フラグは修正されたファームウェアがないことを示す。

【００２７】ここで、ブロックアドレスｎは、予め決められている。また、図２では、修正されたファームウェアのモジュールやジャンプテーブルを格納するために一つのブロックを使用した場合について示しているが、修正されたファームウェアのモジュールやジャンプテーブルを複数のブロックに格納してもよい。

【００２８】図３は、ＲＯＭ２５に格納されているファームウェアの内部構成を示している。

【００２９】ＲＯＭ２５に格納されているファームウェアは、電源オン処理モジュール、初期ジャンプテーブル、初期モジュール１，・・・、初期モジュールｋから構成される。電源オン処理モジュールは、電源オン処理時に実行されるモジュールである。初期ジャンプテーブルは、初期モジュール１，・・・、初期モジュールｋのアドレスを示している。初期モジュール１，・・・、初期モジュールｋは、機能ごとに分割されたプログラムである。

【００３０】図４は、ＲＯＭ２５に格納されている電源オン処理モジュールを処理するマイクロプロセッサのフローチャートである。

【００３１】まず、ステップ１００１において、マイクロプロセッサ２３およびコントローラ２１の初期設定を行う。

【００３２】次いで、ステップ１００２において、ＲＯＭ２５に格納されている初期ジャンプテーブルをＲＡＭ２６にコピーする。図５は、ステップ１００２を処理した後のＲＡＭ２６の内容を示している。

【００３３】ステップ１００３では、フラッシュメモリ２２の修正されたファームウェアが格納されているブロックアドレスｎの修正ファーム管理部２２２をデータバッファ２３に転送している。その後、マイクロプロセッサ２４は、データバッファ２３に転送された修正ファーム管理部２２２の修正ファーム有無フラグをリードし、フラッシュメモリ２２のブロックアドレスｎに修正されたモジュールやジャンプテーブルが格納されているかどうかを確認する（ステップ１００４）。

【００３４】修正されたモジュールやジャンプテーブルがフラッシュメモリ２２のブロックアドレスｎに格納されている場合には、ステップ１００５およびステップ１００６を処理する。ステップ１００５は、フラッシュメモリ２２の修正されたファームウェアが格納されているブロックアドレスｎの修正ファームウェア格納部２２１をデータバッファ２３に転送している。ステップ１００６では、データバッファ２３に転送された修正ファームウェア格納部をＲＡＭ２６にコピーしている。

【００３５】図６は、ステップ１００５およびステップ１００６の処理を行った後のＲＡＭ２６の内容を示している。ここで、ステップ１００２でＲＯＭ２５からコピーされた初期ジャンプテーブルは、フラッシュメモリ２２に格納されていた修正ジャンプテーブルの書き替えられる。ステップ１００６の終了した後は、ステップ１００７に進み、次の処理を行う。

【００３６】修正されたモジュールやジャンプテーブルがフラッシュメモリ２２のブロックアドレスｎに格納されている場合には、ステップ１００７に進み、次の処理を行う。

【００３７】図７は、修正ファームウェア管理部２２２の修正ファーム有無フラグが修正ファームウェアが格納されていないことを示している場合、つまりＲＯＭ２５に格納されているファームウェアに不具合が発生していない場合のファームウェアのフローを図示したものである。ＲＡＭ２６には、ＲＯＭ２５に格納されている初期ジャンプテーブルがコピーされている。

【００３８】図７において、初期モジュール１のモジュールｈを呼び出す個所２５１では、初期ジャンプテーブルの２６１を参照し、モジュールｈのアドレスを得る。２６１は、初期モジュールｈのアドレスを示しているので、初期モジュール１から初期モジュールｈに処理が移る。

【００３９】同様に、初期モジュールｋのモジュールｊを呼び出す個所２５２では、初期ジャンプテーブルの２６２を参照し、モジュールｊのアドレスを得る。２６２は、初期モジュールｊのアドレスを示しているので、初期モジュールｋから初期モジュールｊに処理が移る。

【００４０】図８は、修正ファームウェア管理部２２２の修正ファーム有無フラグが修正ファームウェアが格納されていることを示している場合、つまりＲＯＭ２５に格納されているファームウェアに不具合が発生している場合のファームウェアのフローを図示したものである。また、図８では、ＲＯＭ２５内の初期モジュールｈと初期モジュールｊに不具合が発生したことにより、初期モジュールｈおよび初期モジュールｊに対する修正された修正モジュールｈと修正モジュールｊがＲＡＭ２６に格納されている。さらに、修正モジュールｈと修正モジュールｊのアドレスを変更するために、初期ジャンプテーブルに対する修正された修正ジャンプテーブルもＲＡＭ２６に格納されている。修正ジャンプテーブル、修正モジュールｈおよび修正モジュールｊがＲＡＭ２６へ格納される方法は、図４および図６を用いて説明したように行われる。

【００４１】図８において、初期モジュール１のモジュールｈを呼び出す個所２５１では、修正ジャンプテーブルの２６３を参照し、モジュールｈのアドレスを得る。２６３は、修正モジュールｈのアドレスを示しているので、初期モジュール１から修正モジュールｈに処理が移る。

【００４２】同様に、初期モジュールｋのモジュールｊを呼び出す個所２５２では、修正ジャンプテーブルの２６４を参照し、モジュールｊのアドレスを得る。２６４は、修正モジュールｊのアドレスを示しているので、初期モジュールｋから修正モジュールｊに処理が移る。

【００４３】ここで、図７の初期ジャンプテーブルの２６１と図８の修正ジャンプテーブル２６３はＲＡＭ２６上で同一のアドレスである。同様に、図７の初期ジャンプテーブルの２６２と図８の修正ジャンプテーブル２６４もＲＡＭ２６上で同一のアドレスである。また、マイクロプロセッサ２３は、ＲＡＭ２６に格納されているジャンプテーブルを用いて、モジュールの呼び出しを行う。

【００４４】以上のように、ＲＡＭ２６に格納されたジャンプテーブルを使用することにより、ＲＯＭ２５に格納されているファームウェアに不具合が発生した場合でも、不具合が発生したモジュールをＲＡＭ２６に格納し、さらにジャンプテーブルを修正することで、容易にファームウェアの不具合を対策できる。また、不具合対策のために必要となるＲＡＭ２６の容量は、ジャンプテーブルと不具合が発生したモジュールを格納する容量だけで良いので、ＲＡＭ容量を低減でき、低価格なフラッシュメモリカード２を提供できる。

【００４５】次に、修正されたジャンプテーブルやモジュールをフラッシュメモリ２２に格納するための手順について図９を用いで説明する。図９および図１０は、図４で説明した電源オン処理後のファームウェアを処理す

るマイクロプロセッサ２４にフローチャートである。

【００４６】まず、ステップ２００２において、ホストコンピュータ１がコマンドをフラッシュメモリ２に発行された場合、ステップ２００３でホストコンピュータ１が発行したコマンドを判定する。ステップ２００３において、ホストコンピュータ１が発行したコマンドが特殊ライトコマンドである場合には、ステップ２００５に進み、特殊ライトコマンド以外の場合には、ステップ２００４の通常のコマンド処理を行う。特殊ライトコマンドとは、フラッシュメモリ２２に修正したジャンプテーブルとモジュールを書き込むためのコマンドである。

【００４７】ステップ２００４の通常コマンド処理は、ホストコンピュータ１がユーザデータを書き込んだり、読み出したり、またはフラッシュメモリカード２の情報を読み出したりするためのコマンドであり、ここでは説明を省略する。ステップ２００４が終了後は、ステップ２００２に戻り、ホストコンピュータ１からの次のコマンドを待つ。

【００４８】ステップ２００３で特殊ライトコマンドと判定されると、ステップ２００５において、ホストコンピュータ１からデータバッファ２３へのデータ転送をコントローラ２１に設定する。

【００４９】次いで、ステップ２００６において、ホストコンピュータ１が発行したコマンドに対する準備が整ったことをコントローラ２１に設定し、ホストコンピュータ１からのデータ転送終了を待つ（ステップ２００７）。

【００５０】ホストコンピュータ１からのデータ転送が終了すると、ステップ２００８において、修正ファーム管理部２２２の修正ファーム有無フラグに対応するデータバッファ２３に修正ファームウェア格納部２２１に修正されたジャンプテーブルとモジュールが格納されていることを示すフラグを書き込む。

【００５１】その後、ステップ２００９において、データバッファ２３に書き込まれているホストコンピュータ１からのデータをフラッシュメモリ２２のブロックアドレスｎの修正ファームウェア格納部２２１に、また、データバッファ２３の修正ファーム管理部２２２に対応するデータをフラッシュメモリ２２のブロックアドレスｎの修正ファーム管理部２２２に転送する。

【００５２】フラッシュメモリ２２へのデータ転送終了を待ち（ステップ２０１０）、データ転送が終了した後、ステップ２０１１において、フラッシュメモリ２２にプログラム開始を設定する。

【００５３】ステップ２０１２において、フラッシュメモリ２２へのプログラムが終了した後に、ステップ２００２に戻り、ホストコンピュータ１からの次のコマンドを待つ。

【００５４】ここで、ファームウェアを修正することは、非常に危険な処理である。そこで、フラッシュメモリカード２に制御ピンを設定する。ステップ２００３における特殊ライトコマンドの判定において、前記制御ピンも判定に加える。例えば、グランドレベルを示しているときには、ホストコンピュータ１が特殊ライトコマンドを発行しても、修正ファームウェア格納部２２１および修正ファーム管理部への書き込みを行わないようにする。

【００５５】図４に示した電源オン処理モジュールを実行するマイクロプロセッサ２４のフローチャートの他に実施例について、図１１を用いて説明する。

【００５６】まず、ステップ３００１において、マイクロプロセッサ２３およびコントローラ２１の初期設定を行う。

【００５７】次いで、ステップ３００２では、フラッシュメモリ２２の修正されたファームウェアが格納されているブロックアドレスｎの修正ファーム管理部２２２をデータバッファ２３に転送している。その後、マイクロプロセッサ２４は、データバッファ２３に転送された修正ファーム管理部２２２の修正ファーム有無フラグをリードし、フラッシュメモリ２２のブロックアドレスｎに修正されたモジュールやジャンプテーブルが格納されているかどうかを確認する（ステップ３００３）。

【００５８】修正されたモジュールやジャンプテーブルがフラッシュメモリ２２のブロックアドレスｎに格納されている場合には、ステップ３００４、ステップ３００５そしてステップ３００６を処理する。ステップ３００４は、フラッシュメモリ２２の修正されたファームウェアが格納されているブロックアドレスｎの修正ファームウェア格納部２２１をデータバッファ２３に転送している。ステップ３００５では、データバッファ２３に転送された修正ファームウェア格納部をＲＡＭ２６にコピーしている。ステップ３００６では、ジャンプテーブルをＲＡＭ２６にコピーされた修正ジャンプテーブルに変更するために、ＲＯＭ３５の初期ジャンプテーブルのアドレスに対するオフセットを設定している。オフセットを設定することにより、ＲＯＭ３５の初期ジャンプテーブルの変わりに、ＲＡＭ２６にコピーされた修正ジャンプテーブルを使用することが可能になる。ステップ３００６の後、ステップ３００７に進み、次の処理を行う。

【００５９】修正されたモジュールやジャンプテーブルがフラッシュメモリ２２のブロックアドレスｎに格納されている場合には、ステップ３００７に進み、次の処理を行う。

【００６０】図１２は、修正ファームウェア管理部２２２の修正ファーム有無フラグが修正ファームウェアが格納されていないことを示している場合、つまりＲＯＭ２５に格納されているファームウェアに不具合が発生していない場合のファームウェアのフローを図示したものである。

【００６１】図１２において、初期モジュール１のモジ

ュールｈを呼び出す個所２５１では、初期ジャンプテーブルの２５３を参照し、モジュールｈのアドレスを得る。２５３は、初期モジュールｈのアドレスを示しているので、初期モジュール１から初期モジュールｈに処理が移る。

【００６２】同様に、初期モジュールｋのモジュールｊを呼び出す個所２５２では、初期ジャンプテーブルの２５４を参照し、モジュールｊのアドレスを得る。２５４は、初期モジュールｊのアドレスを示しているので、初期モジュールｋから初期モジュールｊに処理が移る。

【００６３】ここで、修正ファームウェア管理部２２２の修正ファーム有無フラグが修正ファームウェアが格納されていることを示している場合、つまりＲＯＭ２５に格納されているファームウェアに不具合が発生している場合のファームウェアのフローは、図８と同一である。

【００６４】図４に示した電源オン処理モジュールを実行するマイクロプロセッサ２４のフローチャートのさらなる他に実施例について、図１３を用いて説明する。

【００６５】まず、ステップ４００１において、マイクロプロセッサ２３およびコントローラ２１の初期設定を行う。

【００６６】次いで、ステップ４００２では、フラッシュメモリ２２の修正されたファームウェアが格納されているブロックアドレスｎの修正ファーム管理部２２２をデータバッファ２３に転送している。その後、マイクロプロセッサ２４は、データバッファ２３に転送された修正ファーム管理部２２２の修正ファーム有無フラグをリードし、フラッシュメモリ２２のブロックアドレスｎに修正されたモジュールやジャンプテーブルが格納されているかどうかを確認する（ステップ４００３）。

【００６７】修正されたモジュールやジャンプテーブルがフラッシュメモリ２２のブロックアドレスｎに格納されている場合には、ステップ４００４する。ステップ４００４では、ジャンプテーブルをフラッシュメモリ２２に格納された修正ジャンプテーブルに変更するために、ＲＯＭ３５の初期ジャンプテーブルのアドレスに対するオフセットを設定している。オフセットを設定することにより、ＲＯＭ３５の初期ジャンプテーブルの変わりに、フラッシュメモリ２２に格納された修正ジャンプテーブルを使用することが可能になる。ステップ４００４の後、ステップ４００５に進み、次の処理を行う。

【００６８】修正されたモジュールやジャンプテーブルがフラッシュメモリ２２のブロックアドレスｎに格納されている場合には、ステップ４００５に進み、次の処理を行う。

【００６９】図１４は、修正ファームウェア管理部２２２の修正ファーム有無フラグが修正ファームウェアが格納されていることを示している場合、つまりＲＯＭ２５に格納されているファームウェアに不具合が発生している場合のファームウェアのフローを図示したものである。また、図１４では、ＲＯＭ２５内の初期モジュールｈと初期モジュールｊに不具合が発生したことにより、初期モジュールｈおよび初期モジュールｊに対する修正された修正モジュールｈと修正モジュールｊがフラッシュメモリ２２に格納されている。さらに、修正モジュールｈと修正モジュールｊのアドレスを変更するために、初期ジャンプテーブルに対する修正された修正ジャンプテーブルもフラッシュメモリ２２に格納されている。ここで、フラッシュメモリ２２は、ランダムな読み出しが可能なメモリである。

【００７０】図１４において、初期モジュール１のモジュールｈを呼び出す個所２５１では、修正ジャンプテーブルの２２３を参照し、モジュールｈのアドレスを得る。２２３は、修正モジュールｈのアドレスを示しているので、初期モジュール１から修正モジュールｈに処理が移る。

【００７１】同様に、初期モジュールｋのモジュールｊを呼び出す個所２５２では、修正ジャンプテーブルの２２４を参照し、モジュールｊのアドレスを得る。２２４は、修正モジュールｊのアドレスを示しているので、初期モジュールｋから修正モジュールｊに処理が移る。

【００７２】図１３に示すファームウェアの修正方法では、図１に示すＲＡＭ２６が必要なくなるので、さらに低価格なフラッシュメモリカードを提供できる。

【００７３】

【発明の効果】以上説明したように、本発明によれば、フラッシュメモリなどの不揮発性半導体メモリを用いた記憶装置において、内蔵されるファームウェアの修正を容易に行うことができ、低価格な記憶装置を提供できる効果がある。

【図面の簡単な説明】

【図１】本発明の一実施例を示す外部記憶装置の構成を示すブロック図である。

【図２】フラッシュメモリ２２のチップ０における内部構成を示す図である。

【図３】ＲＯＭ２５に格納されているファームウェアの内部構成を示す図である。

【図４】ＲＯＭ２５に格納されている電源オン処理モジュールを処理するマイクロプロセッサのフローチャートである。

【図５】図４のステップ１００２を処理した後のＲＡＭ２６の内容を示す図である。

【図６】図４のステップ１００５およびステップ１００６の処理を行った後のＲＡＭ２６の内容を示す図である。

【図７】修正ファームウェア管理部２２２の修正ファーム有無フラグが修正ファームウェアが格納されていないことを示している場合のファームウェアのフロー図である。

【図８】修正ファームウェア管理部２２２の修正ファー

ム有無フラグが修正ファームウェアが格納されていることを示している場合のファームウェアのフロー図である。

【図９】電源オン処理後のファームウェアを処理するマイクロプロセッサ２４にフローチャートである。

【図１０】電源オン処理後のファームウェアを処理するマイクロプロセッサ２４にフローチャートである。

【図１１】他の実施例である電源オン処理モジュールを実行するマイクロプロセッサ２４のフローチャートである。

【図１２】修正ファームウェア管理部２２２の修正ファーム有無フラグが修正ファームウェアが格納されていないことを示している場合のファームウェアのフロー図である。

【図１３】他の実施例である電源オン処理モジュールを実行するマイクロプロセッサ２４のフローチャートである。

【図１４】修正ファームウェア管理部２２２の修正ファーム有無フラグが修正ファームウェアが格納されていることを示している場合のファームウェアのフロー図である。

【符号の説明】

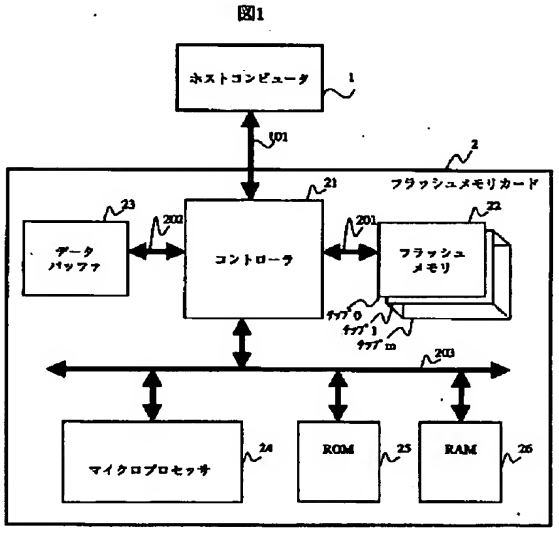
1…ホストコンピュータ、　　2…フラッシュメモリカード（記憶装置）、２１…コントローラ、

２２…フラッシュメモリ、２３…データバッファ、

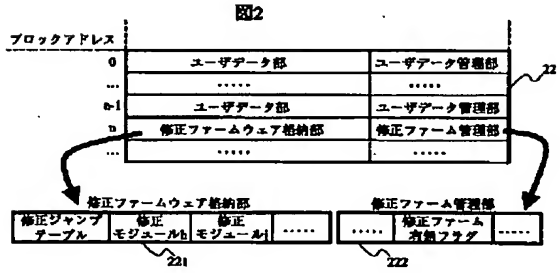
２４…マイクロプロセッサ、２５…ＲＯＭ、

２６…ＲＡＭ。

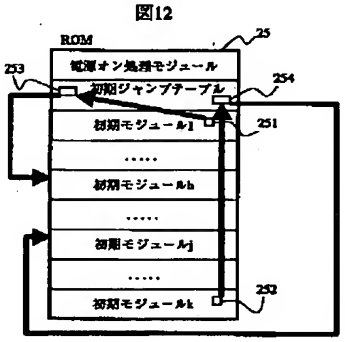
【図１】



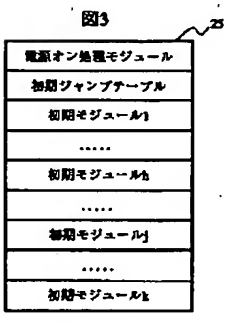

【図２】



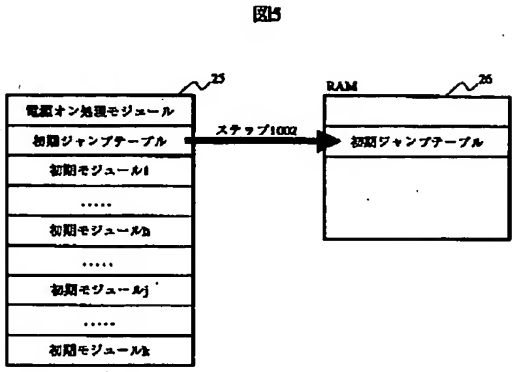

【図１２】




【図３】

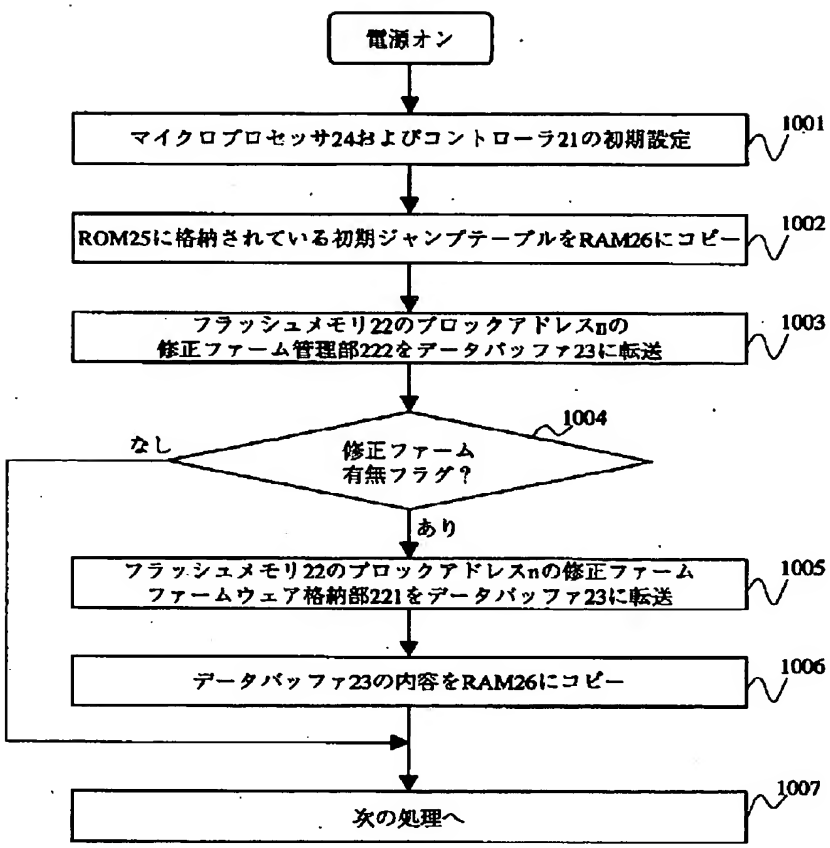



【図５】



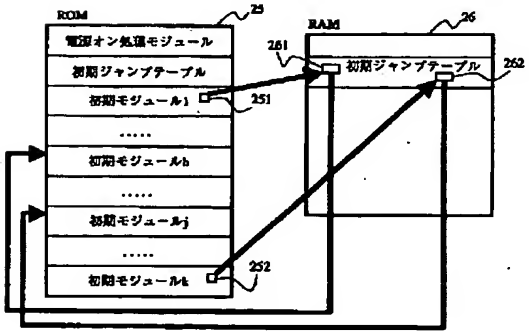
【図4】

図4



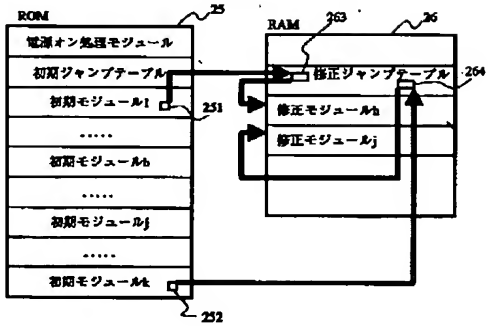

【図7】

図7



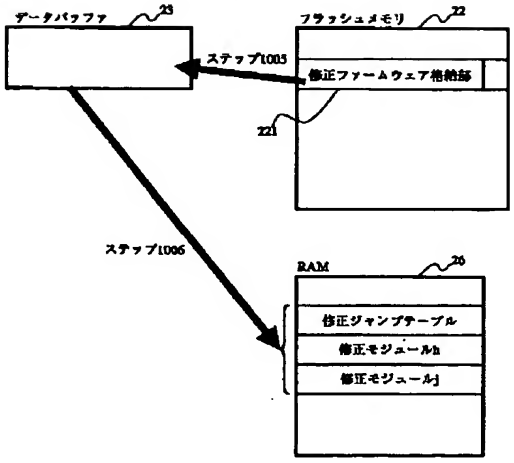

【図8】

図8



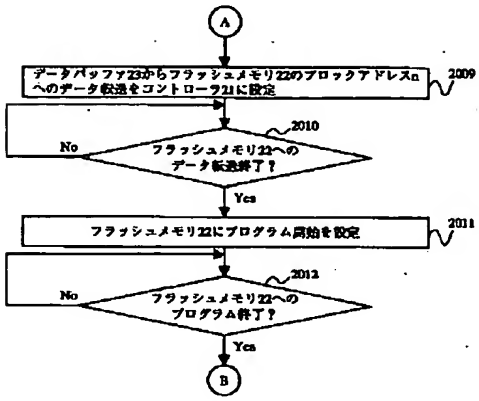
【図6】

図6



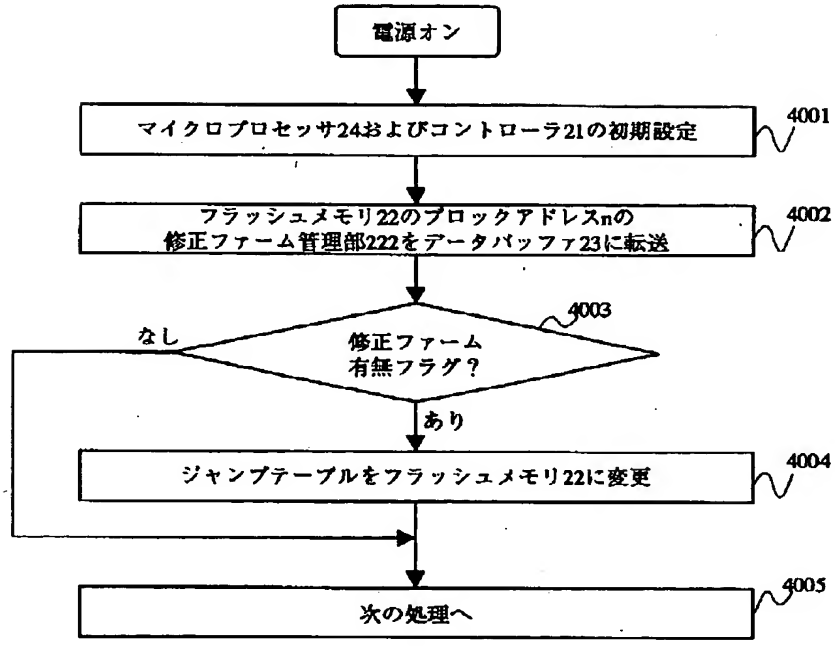

【図10】

図10



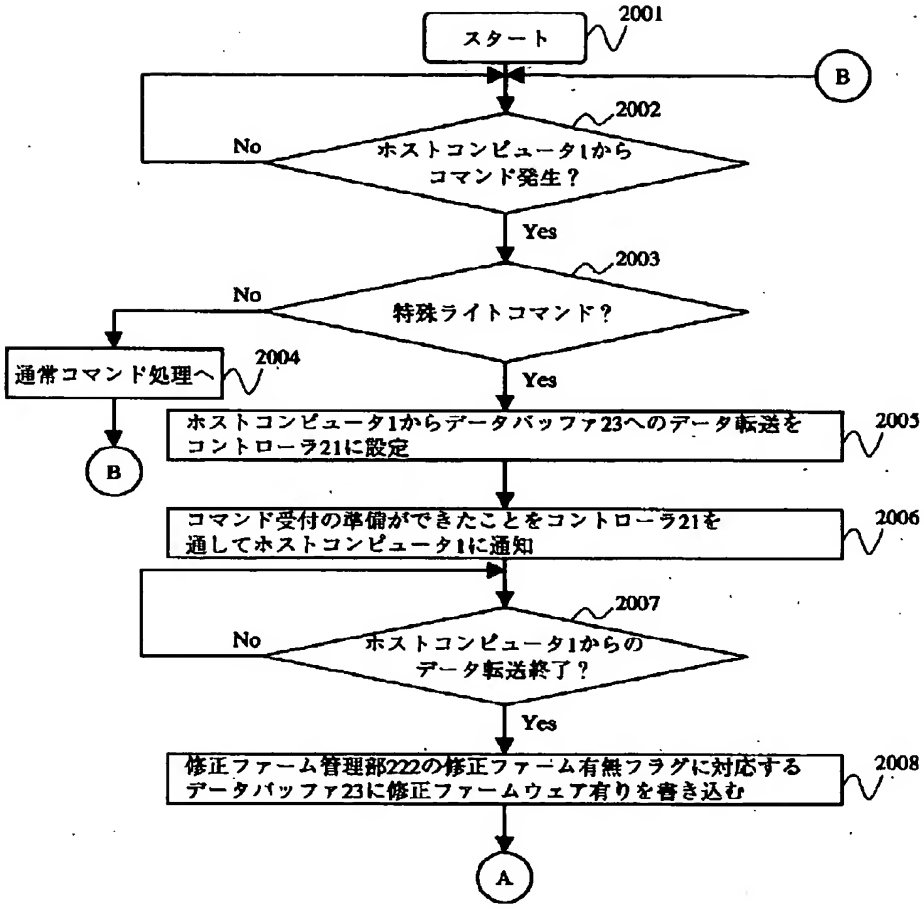

【図13】

図13



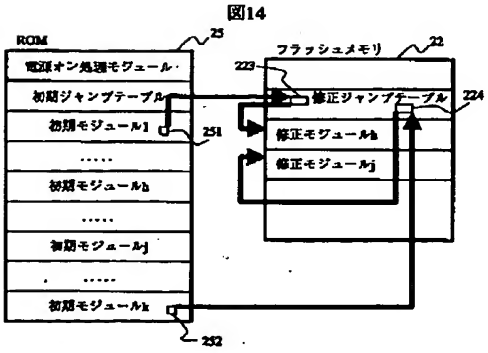
【図9】

図9


スタート
2001
B
2002
No
ホストコンピュータ1から
コマンド発生？
Yes
2003
No
特殊ライトコマンド？
通常コマンド処理へ
2004
Yes
ホストコンピュータ1からデータバッファ23へのデータ転送を
コントローラ21に設定
2005
B
コマンド受付の準備ができたことをコントローラ21を
通してホストコンピュータ1に通知
2006
2007
No
ホストコンピュータ1からの
データ転送終了？
Yes
修正ファーム管理部222の修正ファーム有無フラグに対応する
データバッファ23に修正ファームウェア有りを書き込む
2008
A

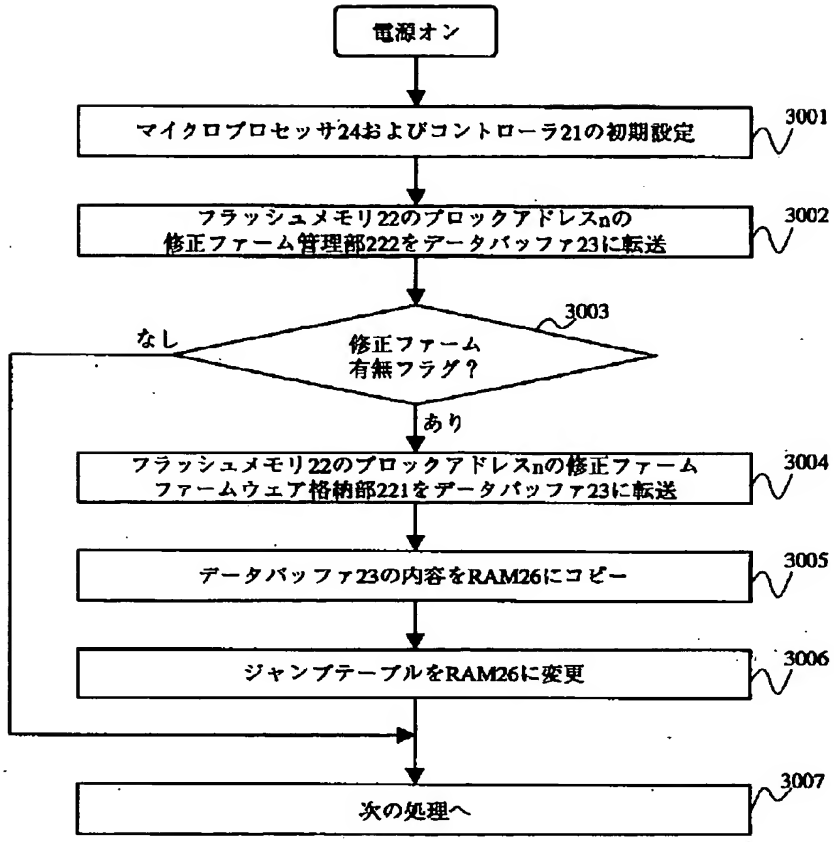
【図14】

図14


ROM
25
電源オン処理モジュール
初期ジャンプテーブル
初期モジュール1
251
初期モジュールk
初期モジュールj
初期モジュールk
252
フラッシュメモリ
22
223
修正ジャンプテーブル
224
修正モジュールk
修正モジュールj

【図11】

図11


電源オン
マイクロプロセッサ24およびコントローラ21の初期設定
3001
フラッシュメモリ22のブロックアドレスnの
修正ファーム管理部222をデータバッファ23に転送
3002
3003
修正ファーム
有無フラグ？
なし
あり
フラッシュメモリ22のブロックアドレスnの修正ファーム
ファームウェア格納部221をデータバッファ23に転送
3004
データバッファ23の内容をRAM26にコピー
3005
ジャンプテーブルをRAM26に変更
3006
次の処理へ
3007

PATENT OFFICE (JP)

Published Unexamined Patent Application(A)
Japanese patent Laid-Open No. Hei 11-265283
Laid-Open Date: September 28, 1999

| Int. Cl.[4] | Identification No. | Reference No. |
|---|---|---|
| G06 9/06 | 540 | G06F 9/06 540E |
| | | 540M |
| 9/22 | 370 | 9/22 370 |
| 12/6 | 520 | 12/06 520E |

*Request for Examination: not requested
No. of claims: 5 (Total 12 pages)

---

Title of the Invention: METHOD FOR CORRECTING FIRMWARE IN MEMORY DEVICE AND MEMORY DEVICE

Japanese Patent Application No. Hei 10-68102
Application Date: March 18, 1998
Inventor:
Takayuki Tamura
c/o System Development Laboratory, Hitachi, Ltd.
1099, Ozenji, Asao-ku, Kawasaki-shi, Kanagawa
Inventor:
Kunihiro Katayama
c/o System Development Laboratory, Hitachi, Ltd.
1099, Ozenji, Asao-ku, Kawasaki-shi, Kanagawa
Inventor:
Kazuo Nakamura
c/o Semiconductor & Integrated Circuits, Hitachi, Ltd.
20-1, Josuihonmachi 5-chome, Kodaira-shi
Applicant:
Hitachi, Ltd.
6, Kanda Surugadai 4-chome, Chiyoda-ku, Tokyo
Agent:
Patent Attorney
Masao Ogawa

(54) TITLE OF THE INVENTION: METHOD FOR CORRECTING FIRMWARE IN MEMORY DEVICE AND MEMORY DEVICE

(57) OBJECT: To provide a low-price memory device that is formed with such a non-volatile semiconductor memory as a flash memory and enables the firmware built in it to be corrected easily. SOLUTION: In a flash memory card 2, instead of an error-occurred firmware module detected in a ROM 25, a corrected firmware module stored in a RAM 26 is used to restore the firmware in the ROM 25 from the error. The corrected firmware module is loaded from the flash memory 22 to the RAM 26 when the flash memory is powered.

WHAT IS CLAIMED IS:

1. A method for restoring firmware stored in a first memory from an error in a memory device comprising a non-volatile semiconductor memory for storing data written from a host computer, a first memory for storing said firmware, and a microprocessor for executing said firmware,

wherein said non-volatile memory stores an error-occurred firmware module and a jump table for denoting a relationship with said error-occurred firmware module when an error occurs in said firmware stored in said first memory,

wherein said error-corrected firmware module and said jump table are loaded from said non-volatile semiconductor memory to a second memory if said non-volatile semiconductor memory stores said error-corrected firmware module and said jump table upon a power on processing, and

wherein said microprocessor refers to said jump table loaded into said second memory to execute said error-corrected firmware module loaded in said second memory instead of said error-occurred firmware module stored in said first memory.

2. The method according to claim 1,

wherein said jump table stored in said first memory is copied into said second memory upon a power on processing,

wherein said microprocessor refers to said jump table

stored in said second memory when said non-volatile semiconductor memory does not store said error-corrected firmware module, thereby executing said firmware stored in said first memory, and

wherein said error-corrected firmware module and said jump table are loaded from said non-volatile semiconductor memory to said second memory when said non-volatile semiconductor memory stores said error-corrected firmware module while said jump table copied from said first memory is overwritten on said loaded jump table, and

wherein said microprocessor refers to said jump table loaded into said second memory to execute said error-corrected firmware module loaded in said second memory instead of said error-occurred firmware module stored in said first memory.

3. A memory device comprising a non-volatile memory for storing data written from a host computer, a first memory for storing firmware, and a microprocessor for executing said firmware,

wherein said memory device further includes a second memory into which an error-corrected firmware module and a jump table are to be loaded from said non-volatile semiconductor memory, and

wherein said memory device executes said error-corrected firmware module stored in said second memory instead of said

error-occurred firmware module stored in said first memory if said non-volatile semiconductor memory stores said error-occurred firmware module and said jump table denoting a relationship with said error-occurred firmware module, as well as said non-volatile semiconductor memory further stores an error-corrected firmware module when an error occurs in said firmware stored in said first memory.

4. A method for correcting error-occurred firmware stored in a first memory comprising a memory device provided with a non-volatile semiconductor memory for storing data written from a host computer, a first memory for storing said firmware, and a microprocessor for executing said firmware,

wherein said microprocessor, when an error occurs in said firmware stored in said first memory, refer to a jump table stored in said non-volatile semiconductor memory to execute an error-corrected firmware module stored in said non-volatile semiconductor memory instead of said error-occurred firmware module stored in said first memory when said non-volatile semiconductor memory stores said error-occurred firmware module and said jump table denoting a relationship with said error-occurred firmware module, and if said non-volatile semiconductor memory further stores an error-corrected firmware module upon a power-on processing.

5. A memory device provided with a non-volatile

semiconductor memory for storing data written from a host computer, a memory for storing firmware, and a microprocessor for executing said firmware;

wherein said memory device, when an error occurs in said firmware, executes an error-corrected firmware module stored in said non-volatile semiconductor memory instead of said error-occurred firmware module stored in said memory if said non-volatile semiconductor memory stores said error-occurred firmware module and said jump table denoting a relationship with said error-occurred firmware module, as well as if said non-volatile semiconductor memory further stores an error-corrected firmware module.

DETAILED DESCRIPTION OF THE INVENTION

[0001]

FIELD OF THE INVENTION

The present invention relates to a memory device that uses an electrically rewritable non-volatile semiconductor memory as a recording medium, more particularly to a memory device formed with a non-volatile semiconductor memory that enables its built-in firmware to be corrected easily.

[0002]

DESCRIPTION OF THE PRIOR ART

There is a memory device formed with such a flash memory as a non-volatile semiconductor memory. The flash memory, when

data is written in a block consisting of one or more sectors, is required to erase the data from the block before the writing. If such a memory device is connected to such a host computer as a personal computer (PC), the memory device is required to analyze commands, etc. issued from the PC to control reading/writing of data requested from the PC.

[0003]

Firmware is stored in such a memory device and used to control this flash memory and processings between host computers. Such a memory device is disclosed, for example, in the official gazette of US Patent No.5606660. In this conventional memory device, the firmware is stored in a non-volatile semiconductor memory and loaded into such a memory as a RAM to be executed by a microprocessor built in the memory device. If the firmware is not stored in the non-volatile semiconductor memory, the subject host computer writes the firmware in the non-volatile semiconductor memory with a special command.

[0004]

[PROBLEMS TO BE SOLVED BY THE INVENTION]

The official gazette of US Patent No.5606660 describes means for storing the subject firmware in a non-volatile semiconductor memory. However, the gazette does not describe any consideration to be taken to correct errors to occur in the firmware. According to the method described in the gazette,

therefore, firmware errors cannot be corrected so easily. This is a problem in the convention memory device. In addition, the RAM capacity increases, since the firmware stored in the non-volatile semiconductor memory is loaded into the RAM. This is another problem in the conventional memory device.

[0005]

Under such circumstances, it is an object of the present invention to make it easy to correct firmware errors even when the errors occur after the firmware is stored in a non-volatile semiconductor memory.

[0006]

It is another object of the present invention to provide a low-price memory device capable of reducing the capacity of the memory used to correct firmware errors.

[0007]

[MEANS FOR SOLVING THE PROBLEMS]

In order to achieve the above objects, according to one aspect of the present invention, the firmware stored in the memory device is configured by a plurality of modules, each corresponding to a function and a jump table to be referred to at the time of calling each module. The jump table stores the address of each firmware module. Furthermore, according another aspect of the present invention, the memory device is configured by a flash memory that is a non-volatile

semiconductor memory, as well as a microprocessor, a ROM for storing the firmware, and a RAM into which the jump table provided in the firmware stored in the ROM or a corrected version of the jump table and a corrected firmware module are loaded. The flash memory stores data written from a host computer, an updated jump table, and a plurality of firmware modules. The ROM stores the firmware beforehand.

[0008]

When the memory device is powered, the jump table stored in the first memory (ROM) is copied into the RAM by the microprocessor. When no error is detected in the firmware, the jump table copied into the RAM denotes the address of each firmware module stored in the ROM as it is.

[0009]

If an error is detected in the firmware stored in the ROM, however, the error is corrected, and then the error-corrected firmware module is stored in the flash memory. At the same time, the jump table is updated so as to denote the address of the error-corrected firmware module, and then stored in the flash memory.

[0010]

If an updated jump table and a plurality of firmware modules are stored in the flash memory after the jump table is copied from the ROM into the RAM, the updated jump table and

the plurality of firmware modules are loaded into the RAM. At this time, the jump table copied from the ROM is overwritten with the updated jump table stored in the flash memory. Consequently, the error-occurred firmware module stored in the ROM is never referred to. Instead, the corrected firmware module loaded in the RAM comes to be referred to.

[0011]

As described above, because both of an error-occurred firmware module and the jump table are corrected/updated before they are stored in the flash memory, the firmware is recovered easily even from errors that occur while it is stored in the ROM.

[0012]

The capacity of the error-occurred module and the jump table is enough to cover the capacity of the RAM required to correct the firmware, so that it is possible to realize a low-price memory device that can reduce the RAM capacity.

[0013]

[DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT]

Hereunder, a description will be made for a memory device composed of a non-volatile semiconductor memory in an embodiment of the present invention with reference to the accompanying drawings.

[0014]

Fig.1 shows a block diagram of a memory device in the embodiment of the present invention. The memory device shown in Fig.1 is employed as an external memory device of a computer.

[0015]

In Fig.1, reference numeral 2 denotes an external memory device used for a host computer 1. The host computer 1 stores/reads data in/from the memory device. For example, the host computer 1 is a personal computer (PC). The external memory device 2 is referred to as a flash memory card, since the flash memory is used as the non-volatile semiconductor memory (memory device).

[16]

The flash memory card 2 sends/receives commands and data to/from the host computer 1 through a standard bus 101. While various interfaces such as the PCMCIA interface and the SCSI interface are usable for the standard bus 101, any of those interfaces can be used for the bus 101 if it uses a predetermined protocol between the host computer 1 and the external memory device required for the host computer 1.

[0017]

The flash memory card 2 includes a controller 21 for interfacing with the host computer 1 and a flash memory 22 and controlling data transfer between itself and the host computer 1/flash memory 22 according to the instruction from the

microprocessor 24.

[0018]

Reference numeral 22 denotes flash memory for storing data written from the host computer 1 and composed of a plurality of flash memory chips. The flash memory 22 stores management information for managing user data handled by the host computer 1, as well as error-corrected firmware modules and an updated jump table and management information used to manage user data.

[0019]

Reference numeral 23 denotes a data buffer for storing data temporarily to be sent/received to/from the host computer 1. The data buffer 23 is also used when the microprocessor 24 reads/writes data from/in the flash memory 22. The data buffer 26 may be any memory such as a static RAM, a dynamic RAM, and a flash memory if it can be accessed at random.

[0020]

Reference numeral 24 denotes a microprocessor that controls commands to be written by the host computer 1, manages the flash memory 22, and controls data transfer between the flash memory 22 and the data buffer 23 and between the host computer 1 and the data buffer 23.

[0021]

Reference numeral 25 denotes a non-volatile memory (ROM) that stores the firmware. The non-volatile memory (ROM) 25 may

be any of a masked ROM, an EEPROM, and a flash memory.

[0022]

Reference numeral 26 denotes a memory (RAM) that can be accessed at random. Corrected firmware modules and the updated jump table are loaded into this memory 26. The memory (RAM) 26 can also be used as a work memory of the microprocessor 24. The RAM 26 may be any of a static RAM, a dynamic RAM, and a flash memory if it can be accessed at random.

[23]

The flash memory 22 is connected to the controller 21 through a flash bus 201. The flash bus 201 has an interface that depends on the flash memory 22. The data buffer 23 is connected to the controller 21 through a data buffer bus 202. The data buffer bus 202 has an interface that depends on the data buffer 23. The controller 21, the microprocessor 24, the ROM 25, and the RAM 26 are connected to each another through a local bus 203. The local bus 203 has an interface that depends on the microprocessor 24.

[0024]

Fig.2 shows an internal block diagram of a chip 0 of the flash memory 22.

[0025]

User data is stored in block addresses 0 to n-1 of the chip 0 respectively. A block address n stores a corrected

firmware module. The block space following the block address n stores information used to manage the flash memory 22. Each user data part includes a user data management part, which retains management information for managing, for example, the state of the user data part. How to use the user data management part will be omitted here.

[0026]

A corrected firmware storage part 221 for storing a corrected version of the firmware also stores the updated jump table and a plurality of error-corrected firmware modules. Similarly to the user data part, the corrected firmware storage part 221 includes a corrected firmware management part 222. The corrected firmware management part 222 retains such information as a corrected firmware presence/absence flag that denotes whether or not any corrected firmware module is stored in the corrected firmware storage part 221. When no error is detected in the firmware stored in the ROM 25, the corrected firmware presence/absence flag denotes that there is no error-corrected firmware module therein.

[0027]

The block address n is predetermined. In Fig.2, a block is used to store error-corrected firmware modules and an updated jump table. However, those error-corrected firmware modules and the updated jump table may also be stored over a plurality

of blocks.

[0028]

Fig.3 shows an internal block diagram of the firmware stored in the ROM 25.

[0029]

The firmware stored in the ROM 25 consists of a power-on processing module, an initial jump table, and initial modules 1 to k. The power-on processing module is executed when the memory device is powered. The initial jump table retains addresses of the initial modules 1 to k. Each of the initial modules 1 to k is a program dedicated to a function.

[0030]

Fig.4 shows a flowchart of the processings performed by the microprocessor 23 for executing the power-on processing module stored in the ROM 25.

[0031]

At first, the microprocessor 23 initializes the microprocessor 23 and the controller 21 in step 1001.

[0032]

The microprocessor 23 then copies the initial jump table stored in the ROM 25 into the RAM 26 in step 1002. Fig.5 shows the contents in the RAM 26 after the processing in step 1002 is completed.

[0033]

In step 1003, the microprocessor 23 transfers the corrected firm management part 222 stored in a block address n to the data buffer 23. The block address n stores a corrected version of the firmware stored in the flash memory 22. After that, the microprocessor 23 reads the corrected firmware presence/absence flag in the corrected firmware management part 222 to check whether or not any error-corrected module and/or updated jump table is stored in the block address n in the flash memory 22 (step 1004).

[0034]

If the check result is YES (stored), the microprocessor 23 executes the processings in steps 1005 and 1006. In step 1005, the microprocessor 23 transfers the corrected firmware storage part 221 stored in the block address n to the data buffer 23. The block address n stores a corrected version of the firmware stored in the flash memory 22. In step 1006, the microprocessor 23 copies the corrected firmware storage part transferred to the data buffer 23 into the RAM 26.

[0035]

Fig.6 shows the contents of the RAM 26 after the processings in steps 1005 and 1006 are completed. The initial jump table copied from the ROM 25 in step 1002 is overwritten with the updated jump table stored in the flash memory 22. After the processing in step 1006, the microprocessor 23 executes the

processing in step 1007.

[0036]

If any error-corrected firmware module and/or jump table is stored in the block address n in the flash memory 22, the microprocessor 23 executes the processing in step 1007.

[0037]

Fig.7 shows a firmware operation flow when the corrected firmware presence/absence flag in the corrected firmware management part 222 denotes that no corrected version of the firmware is stored in the part 222, that is, when no error occurs in the firmware stored in the ROM 25. The RAM 26 stores an initial jump table copied from the ROM 25.

[0038]

In Fig.7, the initial jump table 261 is referred to at a place 251 where a module h of the initial module 1 is called so as to obtain the address of the module h. Reference numeral 261 denotes the address of the initial module h, so that control is passed from the initial module 1 to the initial module h.

[39]

Similarly, at the place 252 where the module j of the initial module k is called, the initial jump table 262 is referred to so as to obtain the address of the module j. Reference numeral 262 denotes the address of the initial module j, so that control is passed from the initial module k to the initial module j.

[0040]

Fig.8 shows a firmware operation flow when the corrected firmware presence/absence flag in the corrected firmware management part 222 denotes that a corrected version of the firmware is stored in the part 222, that is, when an error occurs in the firmware stored in the ROM 25. In Fig.8, an error occurs in each of the initial modules h and j, so that corrected versions of the initial modules h and j are stored in the RAM 26. In addition, an updated version of the initial jump table is also stored in the RAM 26 so as to update the addresses of the corrected initial modules h and j. At this time, the updated jump table and the corrected initial modules h and j are stored in the RAM 26 as described with reference to Figs.4 and 6.

[41]

In Fig.8, at the place 251 where the module h of the initial module 1 is called, the updated jump table 263 is referred to so as to obtain the address of the module h. Reference numeral 263 denotes the address of the corrected module h, so that control is passed from the initial module 1 to the corrected module h.

[0042]

Similarly, at the place 252 where the module j of the initial module k is called, the updated jump table 264 is referred to so as to obtain the address of the module j. Reference numeral 264 denotes the address of the corrected module j, so that control

is passed from the initial module k to the corrected module j.

[0043]

The initial jump table 261 shown in Fig.7 and the updated jump table 263 shown in Fig.8 are stored in the same address here. Similarly, the initial jump table 262 shown in Fig.7 and the updated jump table 264 shown in Fig.8 are stored in the same address. In addition, the microprocessor 23 uses the jump table stored in the RAM 26 to call each module.

[0044]

As described above, because the jump table stored in the RAM 26 is used, if an error occurs in the firmware stored in the ROM 25, the error-occurred firmware module is stored in the RAM 26 and the jump table is updated, thereby the firmware is recovered from the error easily. And, the capacity of the jump table and the error-occurred firmware module is enough to cover the capacity of the RAM 26 required to correct the firmware error. It is thus possible to reduce the RAM capacity and realize a low-price flash memory.

[0045]

Next, a description will be made for a procedure for storing an updated jump table and a corrected firmware module in the flash memory 22 with reference to Fig.9. Figs.9 and 10 show flowcharts of the processings performed by the microprocessor 24 to execute the firmware after the power-on

processing described with reference to Fig.4.

[0046]

At first, if the host computer 1 issues a command to the flash memory 2 in step 2002, the microprocessor 24 checks the command type in step 2003. If the command issued by the host computer 1 is a special write command, the microprocessor 24 goes to step 2005. If not, the microprocessor 24 executes an ordinary command processing in response to the command in step 2004. A special write command means a command for writing both updated jump table and corrected firmware module in the flash memory 22.

[0047]

The commands for ordinary command processings in step 2004 are used by the host computer 1 to write/read user data and read information from the flash memory card 2. The description for them will be omitted here. After the processing in step 2004 is ended, the microprocessor 24 returns to step 2002 to wait for the next command from the host computer 1.

[0048]

If the command issued in step 2003 is a special write command, the microprocessor 24 sets necessary data in the controller 21 so that the host computer 1 transfers data to the data buffer 23 in step 2005.

[0049]

After that, the microprocessor 24 transmits a signal to the controller 21 to denote that the microprocessor 24 is ready to execute a command issued from the host computer 1 in step 2006, then waits for data to be received from the host computer 1 (step 2007).

[0050]

When the data transfer from the host computer 1 is completed, the microprocessor 24 writes a flag in the data buffer 23 corresponding to the corrected firmware presence/absence flag in the corrected firmware management part 222. The flag denotes that both updated jump table and corrected firmware module are stored in the corrected firmware storage part 221.

[0051]

After that, in step 2009, the microprocessor 24 transfers the data received from the host computer and written in the data buffer 23 to the corrected firmware storage part 221 denoted by the block address n of the flash memory 22 and the data corresponding to the corrected firmware management part 222 of the data buffer 23 to the corrected firmware management part 222 denoted by the block address n of the flash memory 22 respectively.

[52]

After the data transfer to the flash memory 22 ends (step 2010), the microprocessor 24 sets the start of the program in

the flash memory 22 in step 2011.

[53]

After the program transfer to the flash memory 22 ends in step 2012, the microprocessor 24 returns to step 2002 to wait for the next command from the host computer 1.

[54]

At this time, firmware correction should be avoided. It can be very dangerous. To avoid this, a control pin is set in the flash memory card 2. The control pin should also be checked in the special command check in step 2003. For example, when the ground level is denoted, it is prevented that a special write command is written in any of the corrected firmware storage part 221 and the corrected firm management part 222 even if the command is issued from the host computer.

[0055]

Next, a description will be made for some other processings that are not included in the flowchart for executing the power-on processing module shown in Fig.4 with reference to Fig.11.

[56]

At first, the microprocessor 23 initializes both of the microprocessor 23 and the controller 21 in step 3001.

[0057]

After that, in step 3002, the microprocessor 24 transfers

the corrected firm management part 222 denoted by the block address n of the flash memory 22 to the data buffer 23. The corrected firmware is stored in the block address n. After that, the microprocessor 24 reads the corrected firm presence/absence flag of the corrected firm management part 222 transferred to the data buffer 23 to check whether or not both corrected firmware module and jump table are stored in the block address n of the flash memory 22 (step 3003).

[0058]

If both corrected firmware module and jump table are stored in the block address n of the flash memory 22, the microprocessor 24 processes the steps 3004 to 3006. In step 3004, the microprocessor 24 transfers the corrected firmware storage part 221 denoted by the block address n in which a corrected version of the firmware stored in the flash memory 22 is stored to the data buffer 23. In step 3005, the microprocessor 24 copies the corrected firmware storage part transferred to the data buffer 23 into the RAM 26. And, in step 3006, the microprocessor 24 sets an offset for the address of the initial jump table in the ROM 35 so as to update the jump table with the updated jump table copied in the RAM 26. This set offset makes it possible to use the updated jump table copied in the RAM 26 instead of the initial jump table stored in the ROM 35. Completing the processing in step 3006, the microprocessor 24

goes to step 3007 to execute the processing.

[0059]

If both corrected firmware module and jump table are stored in the block address n of the flash memory 22, the microprocessor 24 executes the processing in the next step 3007.

[0060]

Fig.12 shows a firmware operation flow when the corrected firm presence/absence flag of the corrected firmware management part 22 denotes that no corrected firmware is stored, that is, when no error occurs in the firmware stored in the ROM 25.

[0061]

In Fig.12, at the place 251 where the module h of the initial firmware module 1 is called, the microprocessor 24 refers to the initial jump table 253 to obtain the address of the module h. Reference numeral 253 denotes the address of the initial module h, so that control goes to the initial module h from the initial firmware module 1.

[0062]

Similarly, at the place 252 where the module j of the initial firmware module k is called, the microprocessor 24 refers to the initial jump table 254 to obtain the address of the module j. Reference numeral 254 denotes the address of the initial module j, so that control goes to the initial module j from the initial firmware module k.

[0063]

If the corrected firm presence/absence flag of the corrected firmware management part 222 denotes that a corrected version of the firmware is stored there, that is, when an error occurs in the firmware stored in the ROM 25, the firmware flow is the same as that shown in Fig.8.

[0064]

Next, a description will be made for still another processing that is not included in the flowchart of the processings by the microprocessor 24 for executing the power-on processing module shown in Fig.4 with reference to Fig.13.

[0065]

At first, in step 4001, the microprocessor 24 initializes both microprocessor 23 and controller 21.

[0066]

After that, in step 4002, the microprocessor 24 transfers the corrected firm management part 222 denoted by the block address n of the flash memory 22 to the data buffer 23. A corrected version of the firmware stored in the flash memory 22 is stored in the block address n. After that, the microprocessor 24 reads the corrected firm presence/absence flag of the corrected firm management part 222 transferred to the data buffer 23 to check whether or not both corrected firmware module and jump table are stored in the block address n of the

flash memory 22 (step 4003).

[0067]

When both corrected firmware module and jump table are stored in the block address n of the flash memory 22, the microprocessor 24 goes to steps 4004. In step 4004, the microprocessor 24 sets an offset for the address of the initial jump table in the ROM 35 so as to update the jump table with the updated jump table stored in the flash memory 22. This set offset makes it possible to use the updated jump table stored in the flash memory 22 instead of the initial jump table stored in the ROM 35. Completing the processing in step 4004, the microprocessor 24 goes to step 4005 to execute the processing.

[0068]

If both corrected firmware module and jump table are stored in the block address n of the flash memory 22, the microprocessor 24 executes the processing in the next step 4005.

[0069]

Fig.14 shows a firmware operation flow when the corrected firm presence/absence flag of the corrected firmware management part 22 denotes that a corrected version of the firmware is stored, that is, when an error occurs in the firmware stored in the ROM 25. In Fig.14, the modules h and j corrected from the initial modules h and j are stored in the flash memory 22, since an error has occurred in each of the initial modules h and j stored in

As described above, according to the present invention, it is possible to provide a low-price memory device composed of such a non-volatile semiconductor memory as a flash memory and enables the built-in firmware to be corrected easily.

[BRIEF DESCRIPTION OF THE DRAWINGS]

Fig.1 is a block diagram of an external memory device in an embodiment of the present invention;

Fig.2 is an internal block diagram of a chip 0 of a flash memory 22;

Fig.3 is an internal block diagram of firmware stored in a ROM 25;

Fig.4 is a flowchart of the processings of a microprocessor for executing a power-on processing module stored in the ROM 25;

Fig.5 is an illustration for denoting the contents in a RAM 26 after the processing in step 1002 in Fig.4 is completed;

Fig.6 is an illustration for denoting the contents in the RAM 26 after the processings in steps 1005 and 1006 in Fig.4 are completed;

Fig.7 is a firmware operation flow when a corrected firm presence/absence flag of a corrected firmware management part 222 denotes that no corrected version of the firmware is stored;

Fig.8 is a firmware operation flow when the corrected firm presence/absence flag of the corrected firmware management part

the ROM 25. In addition, an updated version of the initial jump table is also stored in the flash memory 22 so as to cope with the address changes of the corrected modules h and j. The flash memory 22 enables data stored therein to be read at random.

[0070]

In Fig.14, at the place 251 where the module h of the initial firmware module 1 is called, the microprocessor 24 refers to the updated jump table 223 to obtain the address of the module h. Reference numeral 223 denotes the address of the initial module h, so that control goes to the initial module h from the initial firmware module 1.

[0071]

Similarly, at the place 252 where the module j of the initial firmware module k is called, the microprocessor 24 refers to the initial jump table 224 to obtain the address of the module j. Reference numeral 224 denotes the address of the initial module j, so that control goes to the initial module j from the initial firmware module k.

[0072]

The method for correcting the firmware shown in Fig.13 can omit the RAM 26 shown in Fig.1, so that it is possible to realize a low-price flash memory card.

[0073]

[EFFECT OF THE INVENTION]

222 denotes that a corrected version of the firmware is stored;

Fig.9 is a flowchart of the processings by the microprocessor 24 for executing the firmware after the memory device is powered;

Fig.10 is another flowchart of the processings by the microprocessor 24 for executing the firmware after the memory device is powered;

Fig.11 is a flowchart of the processings by the microprocessor 24 for executing the firmware after the memory device is powered in another embodiment of the present invention;

Fig.12 is a firmware operation flow when the corrected firm presence/absence flag of the corrected firmware management part 222 denotes that no corrected version of the firmware is stored;

Fig.13 is another flowchart of the processings by the microprocessor 24 for executing the power-on processing firmware module in another embodiment of the present invention; and

Fig.14 is a firmware operation flow when the corrected firm presence/absence flag of the corrected firmware management part 222 denotes that a corrected version of the firmware is stored;

[DESCRIPTION OF REFERENCE NUMERALS]

1… Host Computer

2… Flash Memory Card (Memory device)

21… Controller

22… Flash Memory

23… Data Buffer

24… Microprocessor

25… ROM

26… RAM

Fig. 1

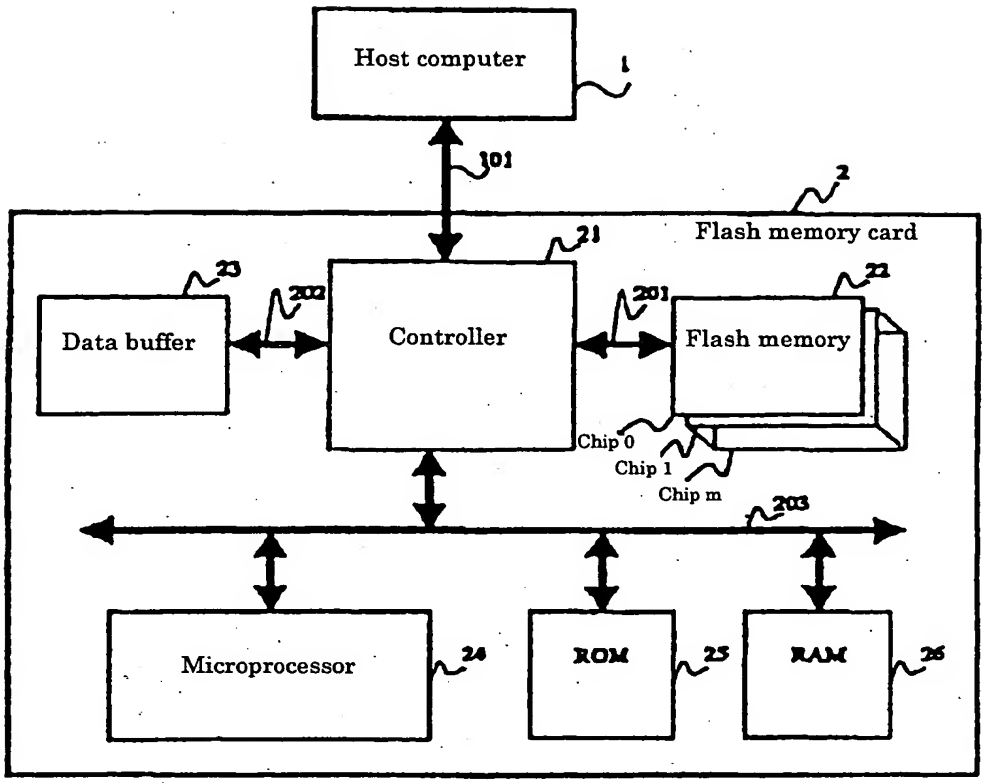

Host computer
1
101
2
Flash memory card
21
23
202
Data buffer
Controller
201
22
Flash memory
Chip 0
Chip 1
Chip m
203
Microprocessor
24
ROM
25
RAM
26

Fig. 2

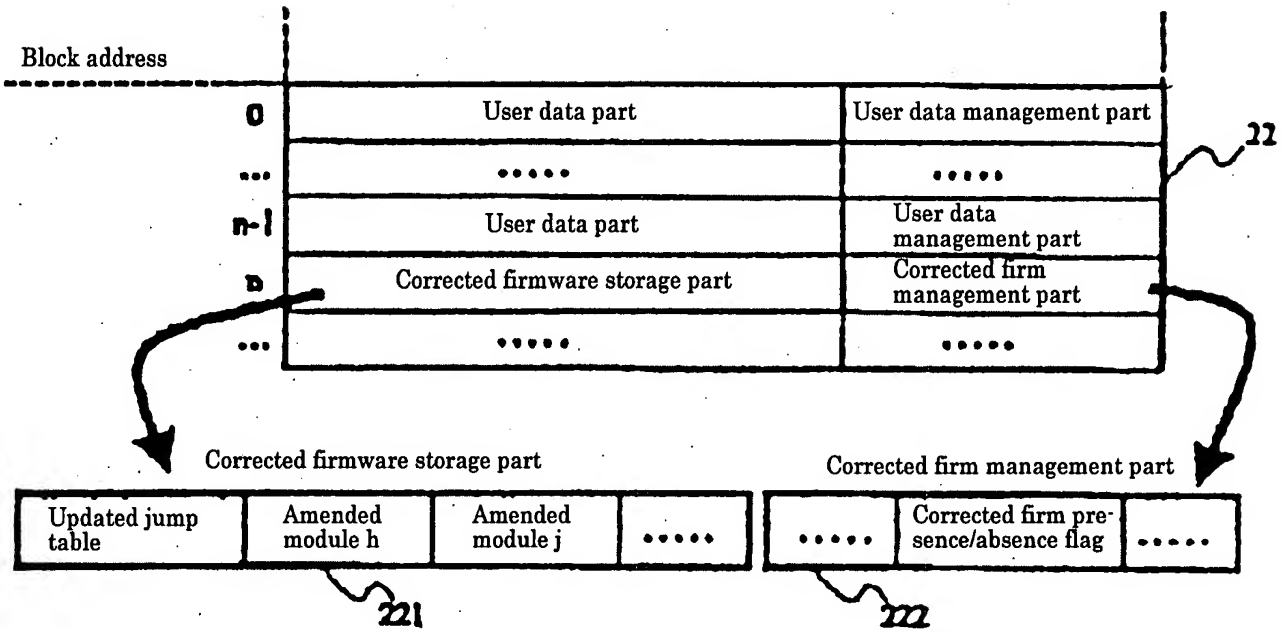

Block address
0
...
n-1
n
...
User data part
User data management part
User data part
User data management part
Corrected firmware storage part
Corrected firm management part
22
Corrected firmware storage part
Corrected firm management part
Updated jump table
Amended module h
Amended module j
Corrected firm presence/absence flag
221
222

Fig. 3

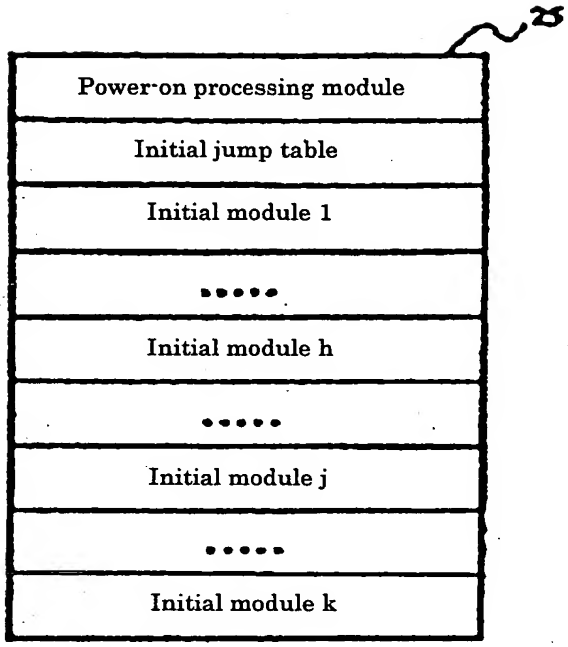

25
Power-on processing module
Initial jump table
Initial module 1
.....
Initial module h
.....
Initial module j
.....
Initial module k

Fig. 4

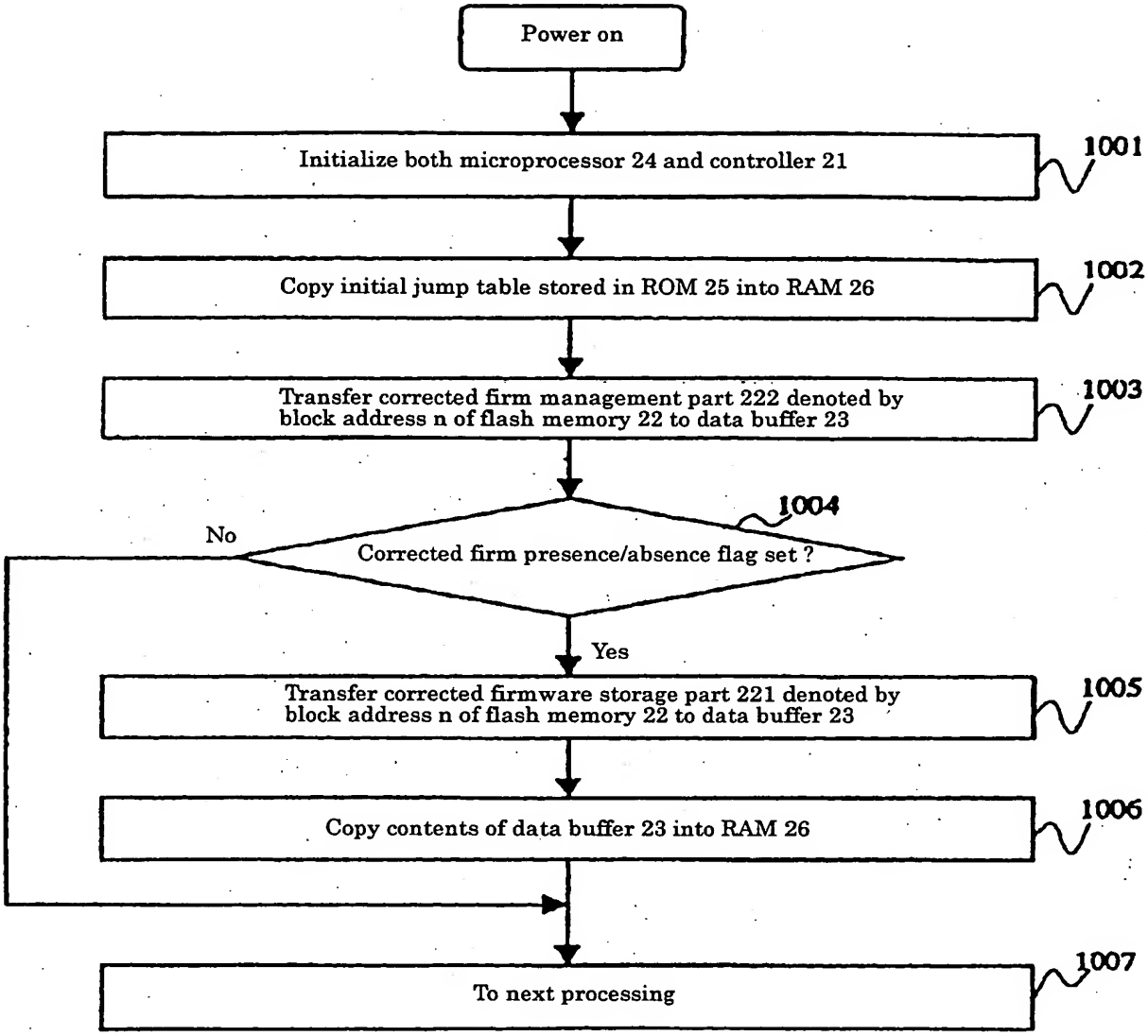

Power on
Initialize both microprocessor 24 and controller 21
1001
Copy initial jump table stored in ROM 25 into RAM 26
1002
Transfer corrected firm management part 222 denoted by block address n of flash memory 22 to data buffer 23
1003
1004
No
Corrected firm presence/absence flag set ?
Yes
Transfer corrected firmware storage part 221 denoted by block address n of flash memory 22 to data buffer 23
1005
Copy contents of data buffer 23 into RAM 26
1006
To next processing
1007

Fig. 5

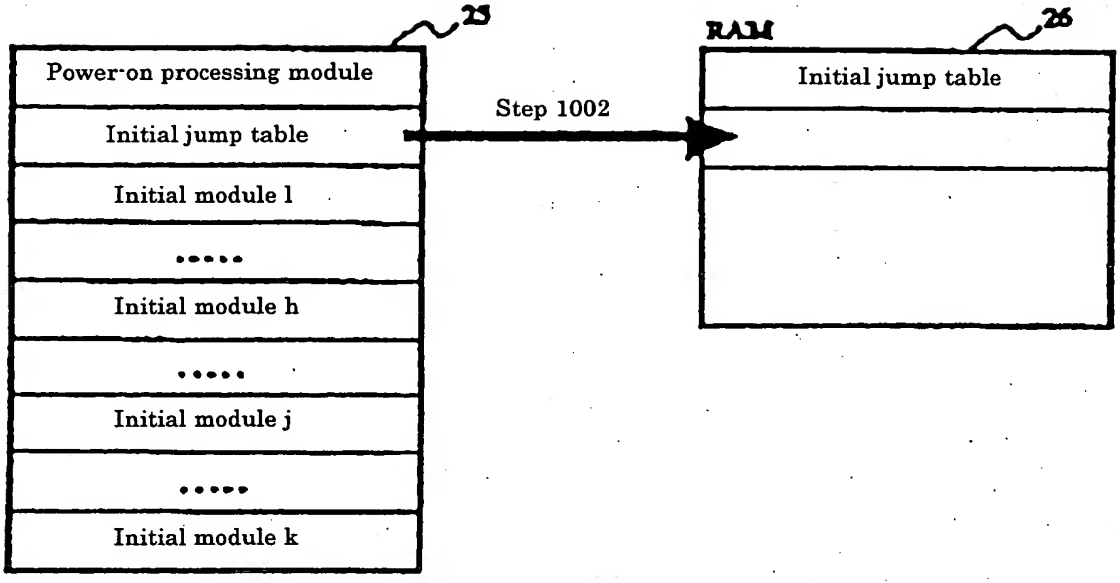

25
Power-on processing module
Initial jump table
Initial module 1
.....
Initial module h
.....
Initial module j
.....
Initial module k
Step 1002
RAM
26
Initial jump table

Fig. 6

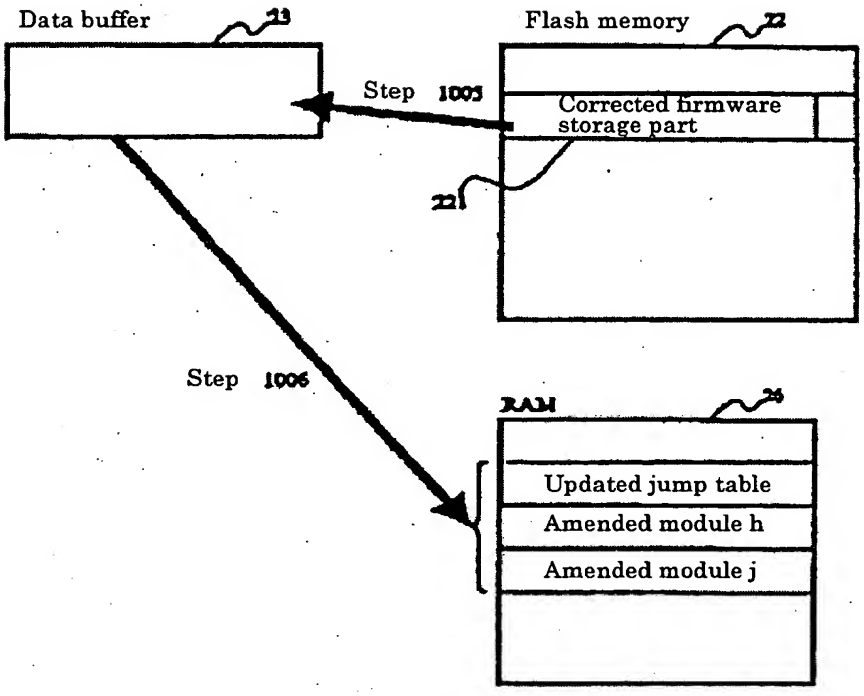

Data buffer
23
Flash memory
22
Step 1005
Corrected firmware storage part
221
Step 1006
RAM
26
Updated jump table
Amended module h
Amended module j

Fig. 7

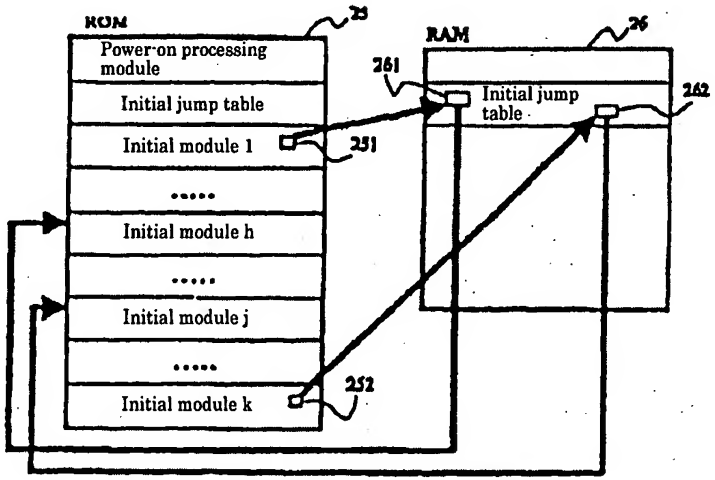

ROM
25
Power-on processing module
Initial jump table
Initial module 1
.....
Initial module h
.....
Initial module j
.....
Initial module k
RAM
26
261
Initial jump table
262
251
252

Fig. 8

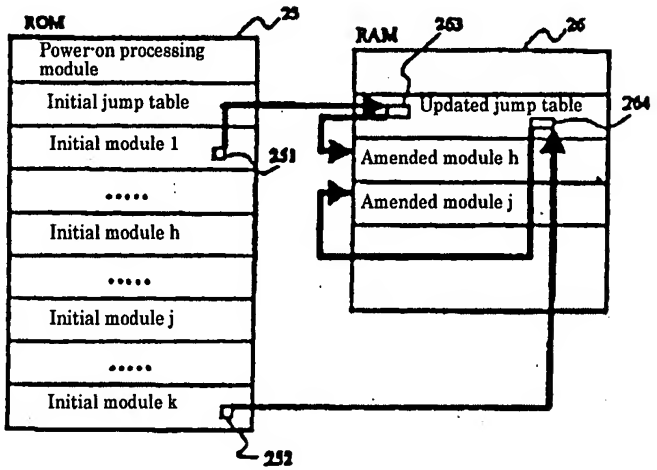

ROM
25
Power-on processing module
Initial jump table
Initial module 1
.....
Initial module h
.....
Initial module j
.....
Initial module k
RAM
263
26
Updated jump table
264
Amended module h
Amended module j
251
252

Fig. 9

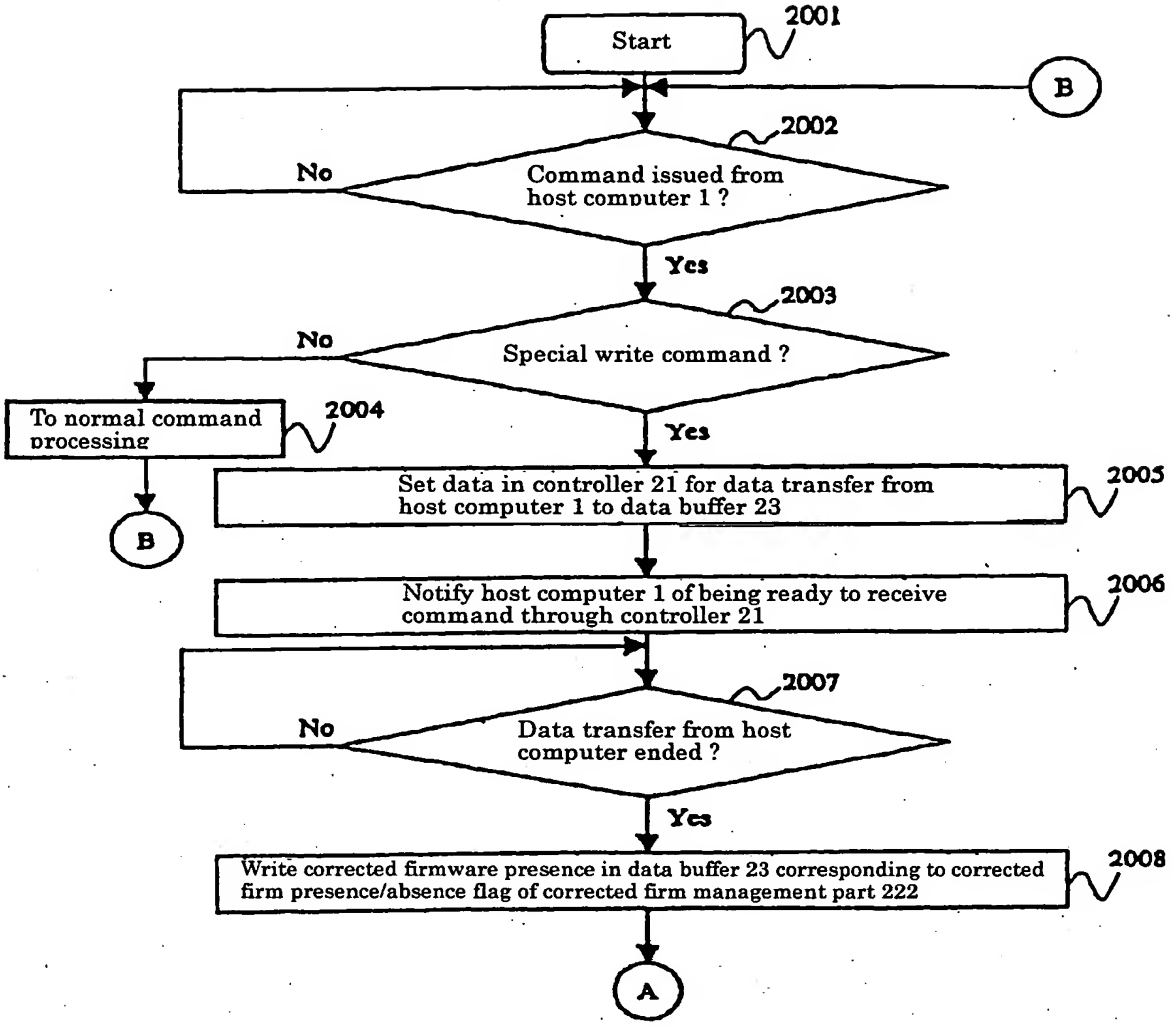

Start
2001
B
2002
No
Command issued from host computer 1 ?
Yes
2003
No
Special write command ?
To normal command processing
2004
Yes
Set data in controller 21 for data transfer from host computer 1 to data buffer 23
2005
B
Notify host computer 1 of being ready to receive command through controller 21
2006
2007
No
Data transfer from host computer ended ?
Yes
Write corrected firmware presence in data buffer 23 corresponding to corrected firm presence/absence flag of corrected firm management part 222
2008
A

Fig. 10

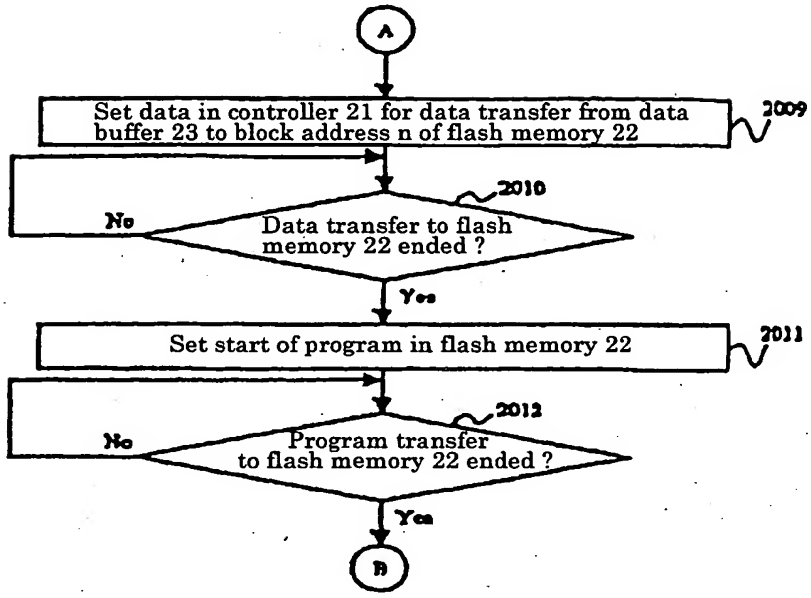

A
Set data in controller 21 for data transfer from data buffer 23 to block address n of flash memory 22
2009
2010
No
Data transfer to flash memory 22 ended ?
Yes
Set start of program in flash memory 22
2011
2012
No
Program transfer to flash memory 22 ended ?
Yes
B

Fig. 11

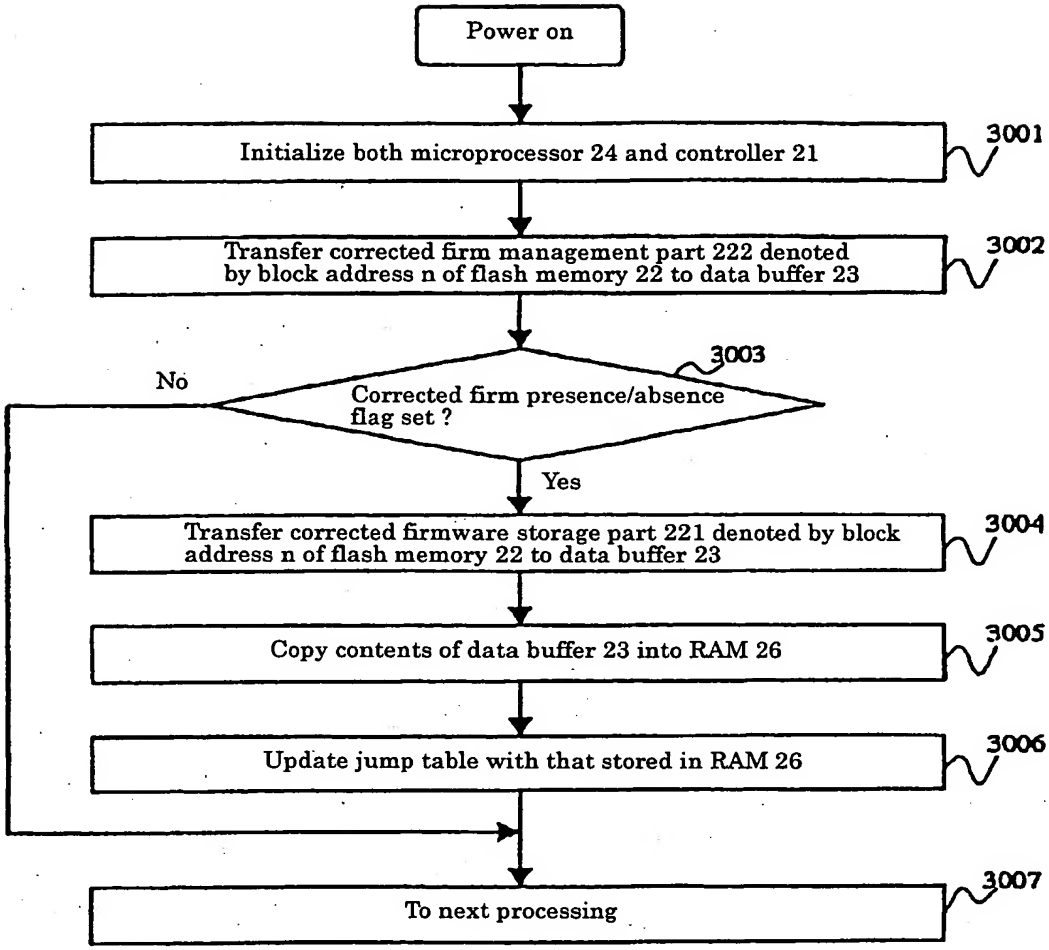

Power on
Initialize both microprocessor 24 and controller 21
3001
Transfer corrected firm management part 222 denoted by block address n of flash memory 22 to data buffer 23
3002
3003
No
Corrected firm presence/absence flag set ?
Yes
Transfer corrected firmware storage part 221 denoted by block address n of flash memory 22 to data buffer 23
3004
Copy contents of data buffer 23 into RAM 26
3005
Update jump table with that stored in RAM 26
3006
To next processing
3007

Fig. 12

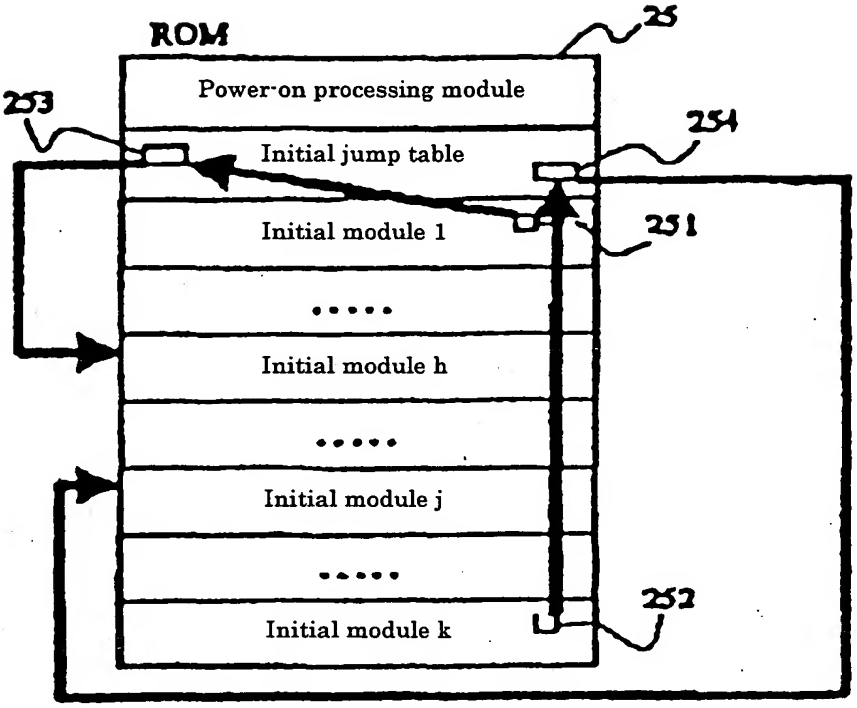

ROM
25
Power-on processing module
253
254
Initial jump table
Initial module 1
251
.....
Initial module h
.....
Initial module j
.....
252
Initial module k

Fig. 13

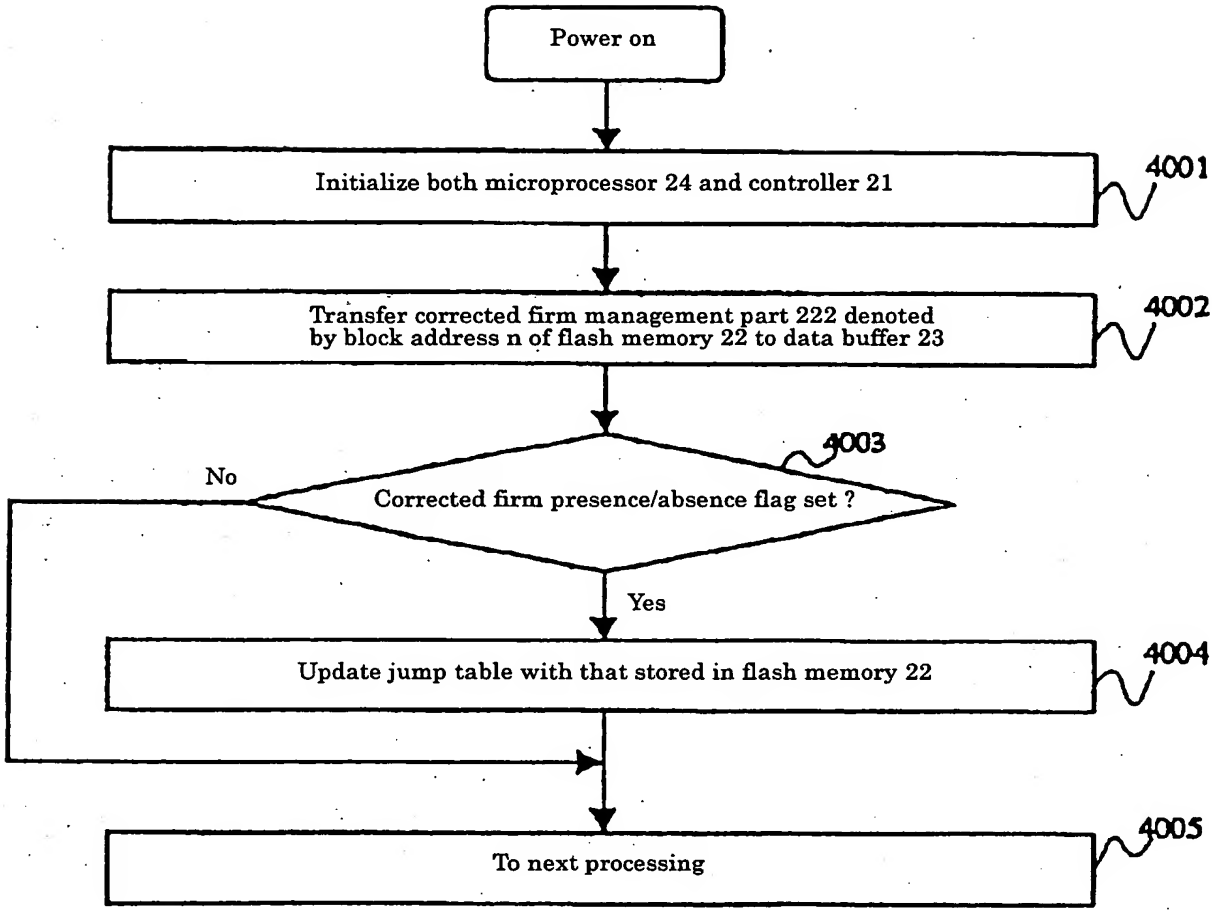
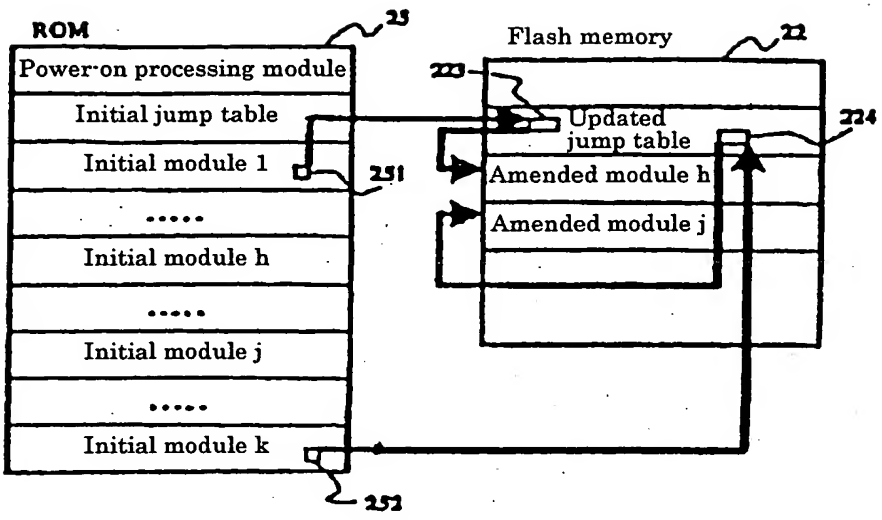

Power on
Initialize both microprocessor 24 and controller 21
4001
Transfer corrected firm management part 222 denoted by block address n of flash memory 22 to data buffer 23
4002
4003
No
Corrected firm presence/absence flag set ?
Yes
Update jump table with that stored in flash memory 22
4004
To next processing
4005

Fig. 14


ROM
25
Power-on processing module
Initial jump table
Initial module 1
.....
Initial module h
.....
Initial module j
.....
Initial module k
251
252
Flash memory
22
223
Updated jump table
224
Amended module h
Amended module j